

*IBM WebSphere Application
Server V6.1 Problem
Determination*

(Course code WA571)

Instructor Guide

ERC 1.0

WebSphere Education

Trademarks

IBM® is a registered trademark of International Business Machines Corporation.

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

AIX®	AS/400®	DB2®
Notes®	Tivoli®	WebSphere®

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

November 2007 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

© Copyright International Business Machines Corporation 2007. All rights reserved.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks	xix
Instructor course overview	xxi
Course description	xxiii
Agenda	xxv
Unit 1. Course introduction	1-1
IBM WebSphere Application Server V6.1 Problem Determination	1-3
Unit objectives	1-5
Introductions	1-7
Course description	1-9
Course prerequisites	1-11
Course objectives	1-13
Course agenda – day 1	1-15
Course agenda – day 2	1-17
Course agenda – day 3	1-19
Course agenda – day 4	1-21
Lab setup	1-23
Network deployment cell topology	1-25
Unit summary	1-27
Unit 2. Overview of WebSphere Application Server components	2-1
Overview of WebSphere Application Server systems and components	2-3
Unit objectives	2-5
WebSphere Application Server architecture	2-7
Web container	2-10
Enterprise JavaBeans (EJB) container	2-14
WebSphere Application Server services	2-17
J2EE Connector Architecture service (JCA)	2-19
Transaction service	2-21
Data Replication Service	2-24
Name service	2-27
Naming topology	2-29
Object Request Broker service	2-32
High availability overview	2-34
Administrative service	2-37
SIBus Web services	2-40
Security service	2-43
Additional services	2-45
WebSphere runtime flow	2-47
Network deployment concepts	2-50
Network deployment runtime flow	2-52

Network deployment administration flow	2-54
File synchronization and file transfer	2-56
Web servers	2-58
Web server plug-ins	2-60
Web server: Managed node (local)	2-62
Typical client request application flow (1 of 2)	2-64
Typical client request application flow (2 of 2)	2-67
Edge Components	2-69
All components in application flow accessible	2-71
Checkpoint (1 of 2)	2-73
Checkpoint (2 of 2)	2-75
Unit summary	2-77
Unit 3. Using the IBM Support Assistant	3-1
Using IBM Support Assistant	3-2
Unit objectives	3-4
IBM Support Assistant	3-6
IBM Support Assistant console	3-8
Updater component	3-10
Installing the plug-ins	3-12
Installing the tools	3-15
Search component	3-17
Performing a search	3-19
Search results	3-21
Product Information component	3-23
Tools component	3-26
IBM Guided Activity Assistant (IGAA)	3-28
Service component	3-30
Data collection scripts	3-33
Creating a portable collector	3-36
Running the portable collector	3-38
Output from Collect Data and Automated Problem Determination	3-40
Unit summary	3-42
Exercise	3-44
Unit 4. Problem determination methodology	4-1
Problem determination methodology	4-2
Unit objectives	4-4
General observations about problem determination	4-6
Types of problem symptoms	4-8
Key steps for problem determination	4-11
Preparation: Before a problem occurs	4-14
Monitoring and problem detection (1 of 3)	4-17
Monitoring and problem detection (2 of 3)	4-19
Monitoring and problem detection (3 of 3)	4-22
System architecture or topology diagram	4-25
Example: Topology diagram	4-27
Establish baselines	4-29

Problem determination artifacts	4-32
Diagnostic data collection plan (1 of 2)	4-35
Diagnostic data collection plan (2 of 2)	4-38
Relief or recovery plan (1 of 2)	4-40
Relief or recovery plan (2 of 2)	4-42
Maintenance plan	4-44
Maintenance strategy: A word about fix packs	4-46
Key steps for problem determination	4-48
Characterize the problem (1 of 2)	4-50
Characterize the problem (2 of 2)	4-52
Key steps for problem determination	4-54
Relief options	4-56
Key steps for problem determination	4-58
"Solve a problem" flow - big picture	4-60
Phase 1: Initial investigation (1 of 2)	4-62
Phase 1: Initial investigation (2 of 2)	4-64
Key steps for problem determination	4-66
"Solve a problem" flow - big picture	4-68
Phase 2: In-depth investigation (1 of 2)	4-70
Phase 2: In-depth investigation (2 of 2)	4-72
Example: Low-level timeline	4-74
For more information	4-76
Checkpoint	4-78
Checkpoint solutions	4-81
Unit summary	4-83
Reference Slides	4-85
Review: Maintenance plan	4-87
Review: Maintenance strategy	4-89
Making backups	4-92
Preventive measures during project life cycle	4-94
Capacity or scalability plan	4-96
Education	4-99
Current architecture plan	4-102
Application design and implementation	4-104
Production traffic profile	4-106
Load or stress testing	4-108
Use a test environment	4-110
Migration plan	4-112
Keep a record of changes	4-115
Facilitate good communication	4-117
Applying APARs	4-119
Unit 5. Problem determination tools and techniques	5-1
Problem determination tools and techniques	5-3
Unit objectives	5-5
Topic objectives	5-7
WebSphere Knowledge Base search	5-9
IBM Support site search	5-12

IBM Software Support tool bar search 5-14

MustGather example 5-16

IBM Guided Activity Assistant (IGAA) 5-18

IGAA layout 5-20

Types of tools and where to find them (1 of 4) 5-23

Types of tools and where to find them (2 of 4) 5-26

Types of tools and where to find them (3 of 4) 5-29

Types of tools and where to find them (4 of 4) 5-31

Tools available externally 5-33

Topic objectives 5-35

Logs and tracing 5-37

Log files and locations 5-39

Viewing logs 5-43

Configuring JVM Logs 5-45

Viewing runtime messages in the console 5-47

HTTP plug-in logs and tracing 5-49

Embedded HTTP Server logs 5-51

Tracing: Differences between V5.x and V6.x 5-53

Traces (1 of 2) 5-55

Enabling trace 5-58

Trace dump and run time 5-61

Setting the Log Detail Level 5-63

Enabling trace by using wsadmin 5-66

Trace header 5-68

Default trace format 5-70

Reading a trace file 5-73

Tools for viewing trace output: Trace Analyzer for WebSphere 5-75

For more information 5-77

Checkpoint 5-79

Checkpoint 5-81

Checkpoint solutions 5-83

Checkpoint solutions 5-85

Unit summary 5-87

Exercise objectives 5-89

Exercise 5-91

Reference Slides 5-93

Log and Trace Analyzer for Java Desktop (LTA-JD) 5-95

Java Logging Architecture - Overview 5-98

Diagnostic Provider infrastructure 5-100

Example Diagnostic Provider scenario 5-103

Unit 6. Introduction to JVM-related problems 6-1

Introduction to JVM-related problems 6-2

Unit objectives 6-4

JVM introduction 6-6

Java virtual machine (JVM) features 6-8

Just-in-time compiler (JIT) basics 6-10

JVM version 6-12

JVM overview	6-15
JVM memory segments	6-17
Understanding garbage collection (GC)	6-20
IBM JDK GC process	6-22
Garbage collection policies	6-25
Generational garbage collection	6-28
Nursery/young generation	6-31
JVM problem determination introduction	6-33
JVM problem determination: Symptom analysis	6-35
JVM problem determination: Data to collect	6-37
Javacore overview	6-39
Javacore file location and naming	6-41
Javacore file subcomponents	6-44
Javacore example (JDK v5)	6-47
Javacore example (JDK 1.4.2 and earlier)	6-49
Verbose GC	6-51
JVM tuning	6-53
Tuning methodology	6-55
Tuning considerations: Heap	6-57
Tuning considerations: Garbage collection	6-59
Tuning considerations: GC policy	6-62
Tuning considerations: Native heap	6-64
JVM tool overview	6-66
Some useful Web addresses and resources	6-68
Related courses and redbooks	6-70
Unit summary	6-72
Unit 7. How to troubleshoot hangs	7-1
How to troubleshoot hangs	7-2
Unit objectives	7-4
Application server hangs	7-6
Application server hang defined	7-8
Thread hangs	7-11
Thread hang examples	7-13
WebSphere process hang detection steps	7-15
How to manually trigger a thread dump	7-18
Thread dump analysis	7-21
Javacore hang indicators	7-23
Javacore hang symptoms	7-26
Hang detection tools	7-28
Topic summary	7-30
Hung thread detection	7-32
WebSphere hung thread detection	7-34
Hung thread detection internals	7-36
Hung thread detection notification	7-38
Hung thread detection false alarms	7-41
Hung thread detection configuration	7-43
Topic summary	7-46

Thread Analyzer	7-48
WebSphere Thread Analyzer	7-50
Thread Analyzer functions	7-52
Thread Analyzer: Summary	7-54
Thread Analyzer: Analysis	7-56
Thread Analyzer: Multiple dump analysis	7-58
Thread Analyzer: Overall thread analysis	7-60
Thread Analyzer: Deadlock indicators	7-63
Thread Analyzer: Overall monitor analysis	7-65
Example output: Servlet thread analysis	7-67
Thread Analyzer: Custom filters	7-69
Topic summary	7-71
Unit summary	7-73
Unit 8. How to troubleshoot crashes	8-1
How to troubleshoot crashes	8-2
Unit objectives	8-4
Debug a crash	8-6
JVM process crash defined	8-8
Crash problem determination: Data to collect	8-11
-Xdump JVM command line argument	8-14
JVM dump initiation: Events	8-17
JVM dump initiation: Types	8-19
JAVA_DUMP_OPTS variable (use now deprecated)	8-21
UNIX operating system common signals	8-23
Windows operating system common signals	8-26
Javacore subcomponents helpful for crash debug	8-28
Javacore example showing a crash	8-31
Javacore fault module	8-33
Javacore current thread details (JDK 1.4.2)	8-36
Steps if crash cause not identified	8-38
Topic summary	8-40
IBM Dump Analyzer for Java	8-42
What is DTFJ?	8-44
Using the DTFJ components - Example	8-46
Where is DTFJ supported?	8-48
Using the Dump Analyzer tool	8-50
Dump Analyzer features	8-52
Dump Analyzer default report contents	8-54
Dump Analyzer: Initialization	8-56
Dump Analyzer: Time estimated	8-58
Dump Analyzer: Analysis completed	8-60
Dump Analyzer deadlock example detail	8-62
Dump Analyzer default report example	8-64
Topic summary	8-66
Unit summary	8-68
Exercise	8-70

Unit 9. Out-of-memory conditions	9-1
Introduction to WebSphere out of memory problems	9-2
Unit objectives	9-4
OutOfMemory error overview	9-6
What is a java.lang.OutOfMemory error?	9-8
JVM heap is too small	9-11
Memory fragmentation (JDK 1.4.2 and earlier)	9-13
Memory leak in the Java code	9-16
Not enough native memory	9-19
Java process restrictions	9-21
The native heap	9-23
Using TPV to anticipate an OutOfMemory error	9-25
Administrative Console PMI settings	9-27
Administrative Console TPV monitoring	9-29
Administrative Console TPV Graph	9-31
TPV usage	9-33
OutOfMemory indicators of the javacore file	9-35
How to obtain a verboseGC log	9-38
Obtaining Java heap dumps	9-41
Use wsadmin to trigger a heap dump	9-43
Java heap dumps with the IBM JDK V5	9-46
Automatic heap dump generation with the IBM JDK V5	9-49
Java heap dumps with the IBM JDK V1.4.2	9-52
How to obtain Java memory dumps on Solaris	9-55
How to obtain a heap dump on HP-UX	9-58
Interpret VerboseGC output	9-60
OutOfMemory: Interpret verboseGC output	9-62
VerboseGC output for optthruput policy	9-64
Error messages output showing dump events	9-67
VerboseGC output for gencon policy	9-69
VerboseGC output for gencon failures	9-72
Extensible Verbose Tool Kit (EVTK)	9-74
Extensible Verbose Toolkit overview	9-76
EVTK usage scenarios	9-78
Plotting data with the EVTK	9-80
Types of graphs	9-82
Garbage collection trigger graphs	9-84
Heap usage and occupancy recommendation	9-86
Reports and recommendations	9-88
IBM Pattern Mapping and Analysis Tool (PMAT)	9-90
PMAT: Summary page	9-92
PMAT: Analysis and recommendations	9-94
PMAT: Chart view	9-96
Analyze Java heap dumps	9-98
OutOfMemory: Interpret heap dumps	9-100
How to analyze a Java heap dump	9-102
Memory Dump Diagnostic Tool for Java (MDD4J)	9-104
MDD4J Analysis Summary tab	9-106

MDD4J Suspects tab 9-108
MDD4J Explore Context and Contents tab 9-110
MDD4J Browse tab 9-112
Checkpoint 9-114
Checkpoint solutions 9-116
Some useful Web addresses 9-118
Unit summary 9-120
Exercise 9-122
Reference JDK 1.4.2 VerboseGC Output Samples 9-124
Sample VerboseGC output: Allocation failure 9-126
How to interpret VerboseGC output 9-128
Sample VerboseGC output: OutOfMemory (1 of 3) 9-130
Sample VerboseGC output: OutOfMemory (2 of 3) 9-132
Sample VerboseGC output: OutOfMemory (3 of 3) 9-134

Unit 10. Database connection and configuration problems 10-1
Introduction to database connection problems 10-2
Unit objectives 10-4
The role of the JDBC provider 10-7
JDBC provider screen 10-10
The role of the data source 10-12
Data source screen (top) 10-15
Creating a database connection: Common problems 10-17
Using the Test Connection service 10-20
Using the Test Connection service (continued) 10-22
JDBC provider configuration problems: Other considerations 10-24
Data source database parameter problems: Identification 10-26
Data source database parameter problems: Diagnosis and resolution 10-28
Data source database authentication problems: Identification 10-30
Data source database authentication problems: Diagnosis and Resolution 10-32
Overview of the DB2 Universal Driver trace facility 10-34
Enabling the DB2 Universal Driver trace facility 10-36
DB2 Universal Driver trace facility output example 10-38
Checkpoint 10-40
Checkpoint solutions 10-42
Unit summary 10-44
Exercise 10-46

Unit 11. Connection pool tuning and management problems 11-1
Connection pool tuning and management problems 11-2
Unit objectives 11-4
What is connection pooling? 11-6
JCA connection pooling architecture 11-8
Types of connection pools in WebSphere 11-11
Sharable versus unshareable connections 11-13
Detecting connection management related problems 11-16
Typical connection pool runtime problem symptoms 11-18
Connection pooling problem determination path 11-20

Troubleshooting connection pool configuration in the problem determination path	11-22
The need for connection pool tuning	11-24
Key connection pool parameters	11-26
Connection pool parameters in the administrative console	11-28
Connection pool tuning tasks	11-31
Monitoring the connection pool using TPV	11-33
TPV connection pool monitoring example	11-36
Generating tuning advice using TPV Performance Advisor	11-38
Performance Advisor tuning advice example	11-41
Tuning the connection pool	11-43
Connection pool tuning best practices	11-46
Troubleshooting connection leaks in the problem determination path	11-49
Connection leaks	11-51
Common causes of connection leaks	11-53
Connection leak diagnosis tasks	11-55
Connection leak trace facility	11-57
Enabling the connection leak trace facility	11-60
What to look for in the trace	11-62
Troubleshooting stale connections in the problem determination path	11-64
Stale connections	11-66
Recovering from a stale connection	11-68
Other stale connection troubleshooting tasks	11-70
Unit summary	11-72
Exercise	11-74
Unit 12. Security configuration-related problems	12-1
Security configuration-related problems	12-2
Unit objectives	12-4
Review of security components and security flows	12-6
Security components overview	12-8
Security flows: Web browser communication	12-11
Authentication flows	12-13
Authorization flows	12-15
Security flows: Java client communication	12-17
Federated repositories	12-19
Federated user repositories	12-21
Common problems and troubleshooting methods	12-23
What can go wrong-The short list	12-25
Approach to troubleshooting security-related issues	12-27
Normal security initialization messages	12-29
Authentication or authorization problem	12-31
Problem related to Secure Sockets Layer (SSL)	12-33
Stack trace in the system log file	12-35
Example: SystemOut.log stack trace	12-37
Example: SystemOut.log exceptions	12-39
Tracing security components	12-41
MustGather tracing requirements	12-43

Result from security components trace12-45

SSL problems and Java 2 security problems12-47

SSL use in WebSphere12-49

How does SSL work?-The “handshake”12-51

SSL client/server identification12-53

SSL problems – Handshake failures (1 of 3)12-55

SSL problems – Handshake failures (2 of 3)12-58

SSL problems – Handshake failures (3 of 3)12-60

SSL error messages12-62

Common SSL connection error message12-64

SSL problem determination steps12-66

Java 2 security problems (1 of 2)12-68

Java 2 security problems (2 of 2)12-70

Administrative console runtime messages, log file messages, and codes12-72

Security Association Service messages12-74

WebSphere Security messages12-76

Web UI Security Center messages12-78

Web Services Security (WS-Security) messages12-80

Virtual member manager (VMM) messages12-82

Additional tools and techniques12-84

Administrative console security PD tools12-86

Security configuration files (1 of 2)12-88

Security configuration files (2 of 2)12-90

Tools to validate the security configuration12-92

Tracking LTPA tokens12-95

Disabling administrative security12-98

Security references12-100

Checkpoint12-102

Checkpoint solutions12-104

Unit summary12-106

Exercise12-108

Unit 13. Application deployment problems 13-1

Application deployment problems13-2

Unit objectives13-4

Application installation methods13-7

Problem areas in application installation13-10

Detecting an application installation issue13-12

Administrative console Troubleshooting menu13-15

Examining console and error log file messages13-17

wsadmin command line exceptions13-20

Typical application installation errors13-22

Typical application start-up errors13-25

Typical application first run errors13-28

Application deployment troubleshooting tools13-30

WebSphere Application Server Toolkit13-33

Resolving an application installation error13-35

Troubleshooting class loader problems13-38

Class loading concepts	13-40
Class loader modes and policies	13-42
Class Loader Viewer example	13-44
Class loading problems (1 of 2)	13-46
Class loading problems (2 of 2)	13-48
How to resolve a ClassNotFoundException problem	13-51
Gathering information for deployment problems	13-53
Checkpoint	13-55
Checkpoint solutions	13-58
Checkpoint solutions (2 of 2)	13-60
Unit summary	13-62
Exercise	13-64
Unit 14. Server start failures	14-1
Server start failures	14-2
Unit objectives	14-4
What happens when a server starts	14-6
Server start messages	14-8
Server boot up process	14-10
Starting a server from the administrative console	14-12
Detecting a failed server start (1 of 2)	14-14
Detecting a failed server start (2 of 2)	14-17
Starting a server from the console—failure	14-20
Common causes of server start failures	14-22
Determining server start issues (1 of 3)	14-24
Determining server start issues (2 of 3)	14-29
Determining server start issues (3 of 3)	14-32
Common configuration error messages	14-34
Resolving server start issues	14-37
Server stop failures	14-39
Administrative Console stop server options	14-41
Stop server by killing the java process	14-43
Checkpoint	14-45
Checkpoint solutions	14-48
Unit summary	14-50
Exercise	14-52
Unit 15. Request flow and Web container problems	15-1
Troubleshooting Web container problems	15-2
Unit objectives	15-4
Web request protocol	15-6
Request flow (HTTP(S))	15-9
Detailed Web request flow	15-12
Typical request methods: GET	15-15
Identifying a working request: Web server access log	15-17
Identifying a working request: Plug-in log (1)	15-20
Identifying a working request: Plug-in log (2)	15-23
IHS and plug-in trace enabling	15-25

Enabling HTTP trace on the application server15-28

Web container diagnostic provider15-30

Troubleshooting a failing request15-32

Preliminary questions (1 of 2)15-34

Preliminary questions (2 of 2)15-37

A word about HTTP response codes15-39

Web server generating a 404 response15-41

Web server generating a 404 response: Solution15-43

Plug-in fails to route a request to application server15-45

Plug-in causing a 404 response: HTTP Plug-in15-47

Application server generating a 404 response15-50

Plug-in generating a 500 response: Solution15-52

HTTP Plug-in 500 level generated15-54

WebSphere Application Server 500 response15-56

Solving a 500 response15-58

Unit summary15-60

Exercise15-62

Unit 16. How to troubleshoot Web services. 16-1

How to troubleshoot Web services16-2

Unit objectives16-4

SOA and Web services16-6

Web services component overview16-8

Web services component interaction overview16-10

Web Services Description Language (WSDL) elements16-12

WebSphere Application Server v6.1 hosting environment for Web services16-15

WebSphere code generation tools16-17

Artifacts created during code generation16-20

Web services security16-23

Checkpoint16-27

Checkpoint solutions16-30

Topic objectives16-32

Web service code generation problems16-36

Web service binding problems16-38

Web service run time problems16-40

Checkpoint16-42

Checkpoint solutions16-44

Topic objectives16-46

Using the IBM Support Assistant Search16-48

Using the MustGathers16-50

Gathering the appropriate data16-53

Automating a Web service MustGather with the IBM Support Assistant16-56

Setting the Web services specific trace strings16-59

Activating trace and reproducing the problem16-62

Configuring tcpmon from the command line16-67

Running tcpmon from IBM Rational Application Developer16-70

Using the Tivoli Performance Viewer to monitor performance problems in Web services
.....16-72

Checkpoint	16-75
Checkpoint solutions	16-77
Topic objectives	16-79
Using RAD and AST to manage the Web service	16-81
Resolving authentication issues in a secure environment	16-84
Resolving interoperability problems caused by incorrect configuration	16-87
Resolving client run time problems	16-90
Exporting the modified Web service	16-92
Redeploying the Web service	16-94
Checkpoint	16-96
Checkpoint solutions	16-98
References	16-100
Unit summary	16-103
Exercise	16-105
Unit 17. Default messaging provider problem determination.	17-1
Default messaging provider problem determination	17-2
Unit objectives	17-4
WebSphere Default Messaging	17-6
WebSphere Default Messaging concept view	17-8
WebSphere Default Messaging concept view	17-10
How to approach a generic problem with WebSphere Default Messaging	17-12
How to approach a generic problem with WebSphere Default Messaging – Step one (1 of 2)	17-14
How to approach a generic problem with WebSphere Default Messaging – Step one (2 of 2)	17-16
How to approach a generic problem with WebSphere Default Messaging – Step two	17-18
How to approach a generic problem with WebSphere Default Messaging – Step three	17-20
How to approach a generic problem with WebSphere Default Messaging – Step four	17-22
How to approach a generic problem with WebSphere Default Messaging – Step four	17-24
How to approach a generic problem with WebSphere Default Messaging – Step five	17-26
How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (1 of 4)	17-28
How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (2 of 4)	17-30
How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (3 of 4)	17-32
How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (4 of 4)	17-34
How to approach a problem with WebSphere Default Messaging – Message flow problems (1 of 3)	17-36
How to approach a problem with WebSphere Default Messaging – Message flow problems (2 of 3)	17-38

How to approach a problem with WebSphere Default Messaging – Message flow problems (3 of 3)17-40

How to approach a problem with WebSphere Default Messaging – Application connection issues17-42

Components and their trace groups (1 of 3)17-44

Components and their trace groups (2 of 3)17-46

Components and their trace groups (3 of 3)17-48

Tracing for WebSphere Default Messaging (1 of 3)17-50

Tracing for WebSphere Default Messaging (2 of 3)17-52

Tracing for WebSphere Default Messaging (3 of 3)17-54

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (1 of 6)17-56

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (2 of 6)17-58

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (3 of 6)17-60

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (4 of 6)17-62

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (5 of 6)17-64

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (6 of 6)17-66

Unit summary17-68

Exercise17-70

Unit 18. WebSphere installation, update, and migration problems 18-1

WebSphere installation, update, and migration problems18-2

Unit objectives18-4

Installing WebSphere Application Server18-6

The launchpad for network deployment18-8

The Profile Management Tool18-10

What can go wrong18-12

Installation problem determination18-14

Common installation problems18-16

Case 1: Launchpad or installation wizard start failure (1)18-18

Case 1: Launchpad or installation wizard start failure (2)18-20

Case 1: Problems starting the launchpad18-22

Case 1: Problems with the installation wizard18-24

Error: Suitable JVM could not be found18-26

Case 2: Installation wizard hangs (1)18-28

Case 2: Installation wizard hangs (2)18-30

Case 2: Data to collect for installation hangs18-32

Case 2: What to look for if installation hangs18-34

Case 2: Identify which process failed (1 of 2)18-36

Case 2: Identify which process failed (2 of 2)18-38

Case 3: Profile creation failure18-40

Case 3: Profile creation failure18-42

Case 3: Verify that profile creation succeeded18-44

Case 3: Profile creation steps	18-46
Case 3: Data to collect if profile creation fails (1 of 3)	18-48
Case 3: Data to collect if profile creation fails (2 of 3)	18-50
Case 3: Data to collect if profile creation fails (3 of 3)	18-53
Case 3: What to look for if profile creation fails (1 of 2)	18-55
Case 3: What to look for if profile creation fails (2 of 2)	18-57
Use ISA to search for profile creation problems	18-60
Case 4: Installation Verification Tool	18-62
Case 4: Installation Verification Tool (1 of 2)	18-64
Case 4: Installation Verification Tool (2 of 2)	18-66
Case 4: Data to collect if IVT fails	18-68
Case 4: What to look for if IVT fails (1 of 3)	18-70
Case 4: What to look for if IVT fails (2 of 3)	18-72
Case 4: What to look for if IVT fails (3 of 3)	18-74
Installation Verification Utility	18-76
Installation Verification Utility: installver	18-78
Run the verifyinstallver command	18-81
Running the installver utility	18-83
Creating a new baseline checksum	18-85
Uninstalling after failed or hung install	18-88
Recovering from a failed or hung installation (1 of 2)	18-90
Recovering from a failed or hung installation (2 of 2)	18-92
Reinstall with tracing turned on	18-96
Applying maintenance updates	18-98
Verify current version levels and applied updates	18-100
Installing maintenance packages (1 of 2)	18-102
Installing maintenance packages (2 of 2)	18-104
Troubleshooting migration problems	18-106
Migrating an earlier version of WebSphere version 6.1.	18-108
Migration wizard	18-110
Troubleshooting migration (1 of 2)	18-112
Troubleshooting migration (2 of 2)	18-114
Before contacting IBM support	18-116
Checkpoint	18-118
Checkpoint solutions	18-120
Unit summary	18-122
Exercise	18-124
Unit 19. Course summary	19-1
Course summary	19-2
Course summary	19-4
Resources	19-6
Resources (continued)	19-8
Articles and white papers	19-10
Articles and white papers (continued)	19-12
Other WebSphere courses	19-14
Evaluations	19-16

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

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

AIX®

AS/400®

DB2®

Notes®

Tivoli®

WebSphere®

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

Instructor course overview

This course will teach WebSphere administrators skills needed to troubleshoot problems, differentiate application and infrastructure problems, and to work with IBM Support teams to report, isolate, identify, and resolve WebSphere infrastructure problems.

This course will help customers manage WebSphere Application Server related problems more skillfully within their organizations and enable them to access online support assistant tools to resolve problems. It will also help them to communicate with IBM Support teams more effectively in order to identify the problem and find a solution. This course will teach students methodologies and techniques for problem determination in addition to discussing some common scenarios that they may face in their daily activities associated with WebSphere Application Server.

Course strategy

The course strategy is to teach students the objective skills through lectures and hands-on labs.

Summary of changes in this edition

WA571 is an updated version of SW570 for WebSphere Application Server V6.1. Following are the major changes in the new course:

- The course is now 4 days in duration
- The following units have been added:
 - Unit 16 - How to troubleshoot Web services
 - Unit 17 - Default messaging provider problem determination
- The following exercises have been added:
 - Exercise 2 - Introduction to problem determination tools
 - Exercise 11 - Troubleshooting Web services
 - Exercise 12 - Troubleshooting the default messaging provider
- The security configuration-related problem unit has been expanded to include additional topics on SSL and Java 2 security
- The installation problem determination unit has been enhanced to include maintenance updates and migration issues
- The order of units has been re-arranged

- The JVM units have been moved to day 1 and day 2. These units include:
 - Unit 6 - Introduction to JVM-related problems
 - Unit 7 - How to troubleshoot hangs
 - Unit 8 - How to troubleshoot crashes
 - Unit 9 - Out-of-memory conditions
- The installation, update, and migration problems unit is now at the end of the course

Course description

IBM WebSphere Application Server V6.1 Problem Determination

Duration: 4 days

Purpose

This course will help customers manage WebSphere Application Server related problems more skillfully within their organizations and enable them to access online support assistant tools to resolve problems. It will also help them to communicate with IBM Support teams more effectively in order to identify the problem and find a solution. This course will teach students methodologies and techniques for problem determination in addition to discussing some common scenarios that they may face in their daily activities associated with WebSphere Application Server.

Audience

WebSphere Administrators, business partners, ISVs who work with WebSphere related applications, and consultants who work on WebSphere related projects.

Prerequisites

Students must have basic operating skills for Windows and/or UNIX operating systems. In addition, students must have WebSphere administration skills acquired by having completed a WebSphere Application Server Administration course such as WA361 or gained through practical experience administering a WebSphere Application Server environment.

Objectives

After completing this course, you should be able to:

- Apply problem determination techniques
- Work with problem determination tools to isolate, identify, and resolve problems
- Identify frequently faced problems and resolve them
- Use IBM Support Assistant to help identify solutions
- Work and communicate more effectively with IBM Support teams

Curriculum relationship

- WA361
- WF881

Agenda

Day 1

- (00:30) Unit 1 - Course introduction
- (00:45) Unit 2 - Overview of WebSphere Application Server components
- (00:30) Unit 3 - Using the IBM Support Assistant
- (00:45) Exercise 1 - A guided tour of IBM Support Assistant
- (01:00) Unit 4 - Problem determination methodology
- (00:45) Unit 5 - Problem determination techniques and tools
- (00:45) Exercise 2 - Introduction to problem determination tools
- (01:00) Unit 6 - Introduction to JVM-related problems

Day 2

- (00:30) Unit 7 - How to troubleshoot hangs
- (00:30) Unit 8 - How to troubleshoot crashes
- (00:45) Exercise 3 - Hung threads and server crash detection
- (00:45) Unit 9 - Out-of-memory conditions
- (01:15) Exercise 4 - Troubleshoot an out-of-memory condition
- (00:30) Unit 10 - Database connection and configuration problems
- (01:00) Exercise 5 - Troubleshooting database configuration problems

Day 3

- (00:45) Unit 11 - Connection pool tuning and management problems
- (00:45) Exercise 6 - Troubleshoot a connection pool problem
- (01:00) Unit 12 - Security configuration-related problem
- (01:00) Exercise 7 - Troubleshooting security problems
- (00:30) Module 13 - Application deployment problems
- (01:00) Exercise 8 - Troubleshooting problems with applications
- (00:30) Module 14 - Server start failures
- (01:00) Exercise 9 - Troubleshooting server start failures

Day 4

- (01:00) Unit 15 - Request flow and Web containers problems
- (00:45) Exercise 10 - Troubleshooting request flow and Web containers problems
- (00:45) Unit 16 - How to troubleshoot Web services
- (00:45) Exercise 11 - Troubleshooting Web services
- (00:45) Unit 17 - Default messaging provider problem determination
- (00:45) Exercise 12 - Troubleshooting the default messaging provider

(00:45) Unit 18 - Installation, update, and migration problems
(00:45) Exercise 13 - Troubleshooting installation, update, and migration problems
(00:10) Unit 19 - Course summary

Unit 1. Course introduction

Estimated time

00:30

What this unit is about

This unit describes the objectives and agenda for the course.

What you should be able to do

After completing this unit, you should be able to:

- Describe the purpose of the course
- Describe the prerequisites
- List the objectives
- Describe the agenda for the course

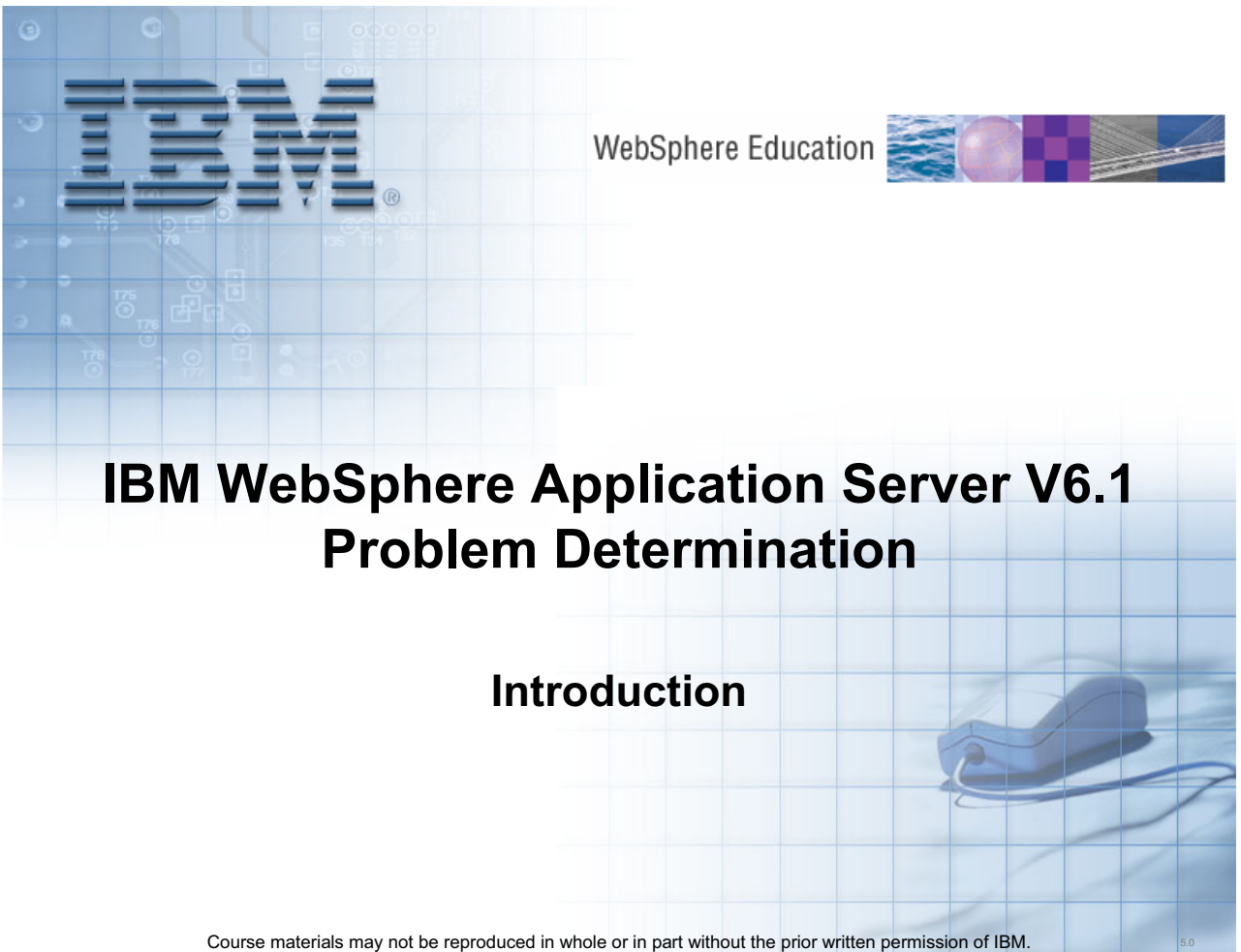


Figure 1-1. IBM WebSphere Application Server V6.1 Problem Determination

WA5711.0

Notes:

Instructor notes

What students will do —

How students will do it —

What students will learn —

How this will help students on their job —

Unit objectives

After completing this unit, you should be able to:

- Describe the purpose and scope of the course
- List the course objectives
- Describe the course prerequisites
- Describe the agenda for the course

Figure 1-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Introductions

- Instructors
- Students
 - Introduce yourself
 - Name and organization
 - Java and Java enterprise knowledge
 - IBM WebSphere Application Server knowledge
 - Internet and Web tools you use
 - What you hope to get from this class

Figure 1-3. Introductions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course description

- Title: IBM WebSphere Application Server V6 Problem Determination
- Course code: WA571
- Duration: 4 days
- Purpose:
 - This course will help customers manage WebSphere-related problems more skillfully within their organizations and enable them to access online support assistant tools to resolve problems.
 - If all else fails, they can communicate with IBM support teams more effectively in order to help identify the problem and find a solution.

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course prerequisites

- To get the most out of this course offering, students should meet the following prerequisites. Students who do not meet these prerequisites may not be able to fully understand and utilize the materials presented in the course.
 - An understanding of basic Internet concepts
 - Experience using a Web browser
 - Basic operational skills for the Windows operating system
 - WA361 or equivalent skills in WebSphere administration

Figure 1-5. Course prerequisites

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course objectives

After completing this course, you should be able to:

- Apply problem determination techniques
- Use problem determination tools to isolate, identify, and resolve problems
- Identify frequently faced problems and resolve them
- Use IBM Support Assistant to help identify solutions
- Communicate with IBM support teams more effectively

Figure 1-6. Course objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course agenda – day 1

- Unit 1: Course introduction
- Unit 2: Overview of WebSphere Application Server systems and components
- Unit 3: Using IBM Support Assistant
 - Exercise 1: A guided tour of using IBM Support Assistant
- Unit 4: Problem determination methodology
- Unit 5: Problem determination techniques and tools
 - Exercise 2: Introduction to problem determination tools
- Unit 6: Introduction to JVM-related problems

Figure 1-7. Course agenda – day 1

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course agenda – day 2

- Unit 7: How to troubleshoot hangs
- Unit 8: How to troubleshoot crashes
 - Exercise 3: Hung threads and server crash detection
- Unit 9: Out-of-memory conditions
 - Exercise 4: Troubleshooting an out-of-memory condition
- Unit 10: Database connection and configuration problems
 - Exercise 5: Troubleshooting database configuration problems

Figure 1-8. Course agenda – day 2

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course agenda – day 3

- Unit 11: Connection pool tuning and management problems
 - Exercise 6: Troubleshoot a connection pool problem
- Unit 12: Security configuration-related problems
 - Exercise 7: Troubleshooting security problems
- Unit 13: Application deployment problems
 - Exercise 8: Troubleshooting application problems
- Unit 14: Server start failures
 - Exercise 9: Troubleshooting server start failures

Figure 1-9. Course agenda – day 3

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course agenda – day 4

- Unit 15: Request flow and Web container problems
 - Exercise 10: Troubleshooting request flow and Web container problems
- Unit 16: How to troubleshoot Web services
 - Exercise 11: Troubleshooting Web services
- Unit 17: Default messaging provider problem determination
 - Exercise 12: Troubleshooting the default messaging provider
- Unit 18: Installation, update, and migration problems
 - Exercise 13: Troubleshooting installation problems
- Unit 19: Course summary

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Lab setup

- Course is taught at WebSphere Application Server Network Deployment V6.1 product level
- Screen captures are on Windows
- Lab machines: Each lab machine has a VMWare image installed and configured as follows:
 - Deployment manager profile
 - Profile1-node agent and server1
 - IBM HTTP server profile (unmanaged)-webserver1
 - IBM Tivoli Directory Server
 - WebSphere Application Server Toolkit

Figure 1-11. Lab setup

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Network deployment cell topology

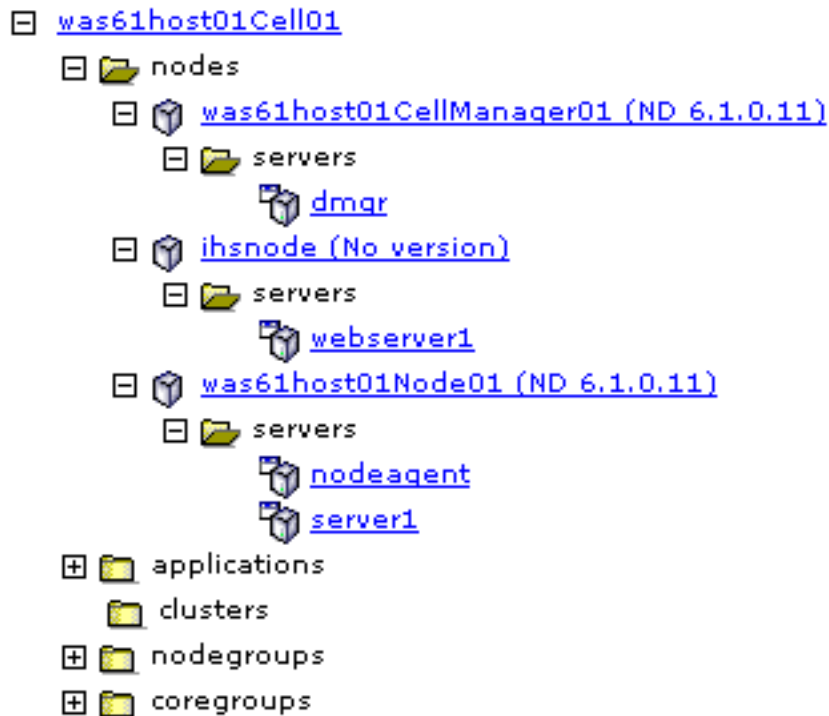


Figure 1-12. Network deployment cell topology

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

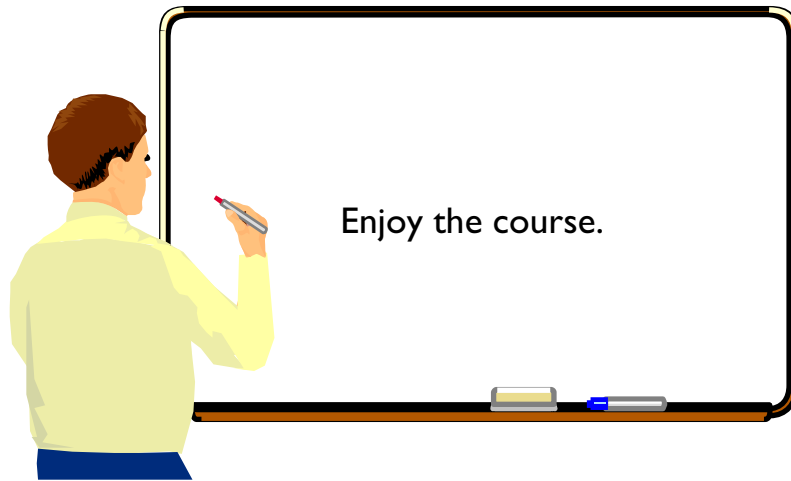


Figure 1-13. Unit summary

WA5711.0

Notes:

Unit 2. Overview of WebSphere Application Server components

Estimated time

00:45

What this unit is about

This module provides an overview of the topology of a system and identifies and describes key components as well as troubleshooting points.

What you should be able to do

After completing this unit, you should be able to:

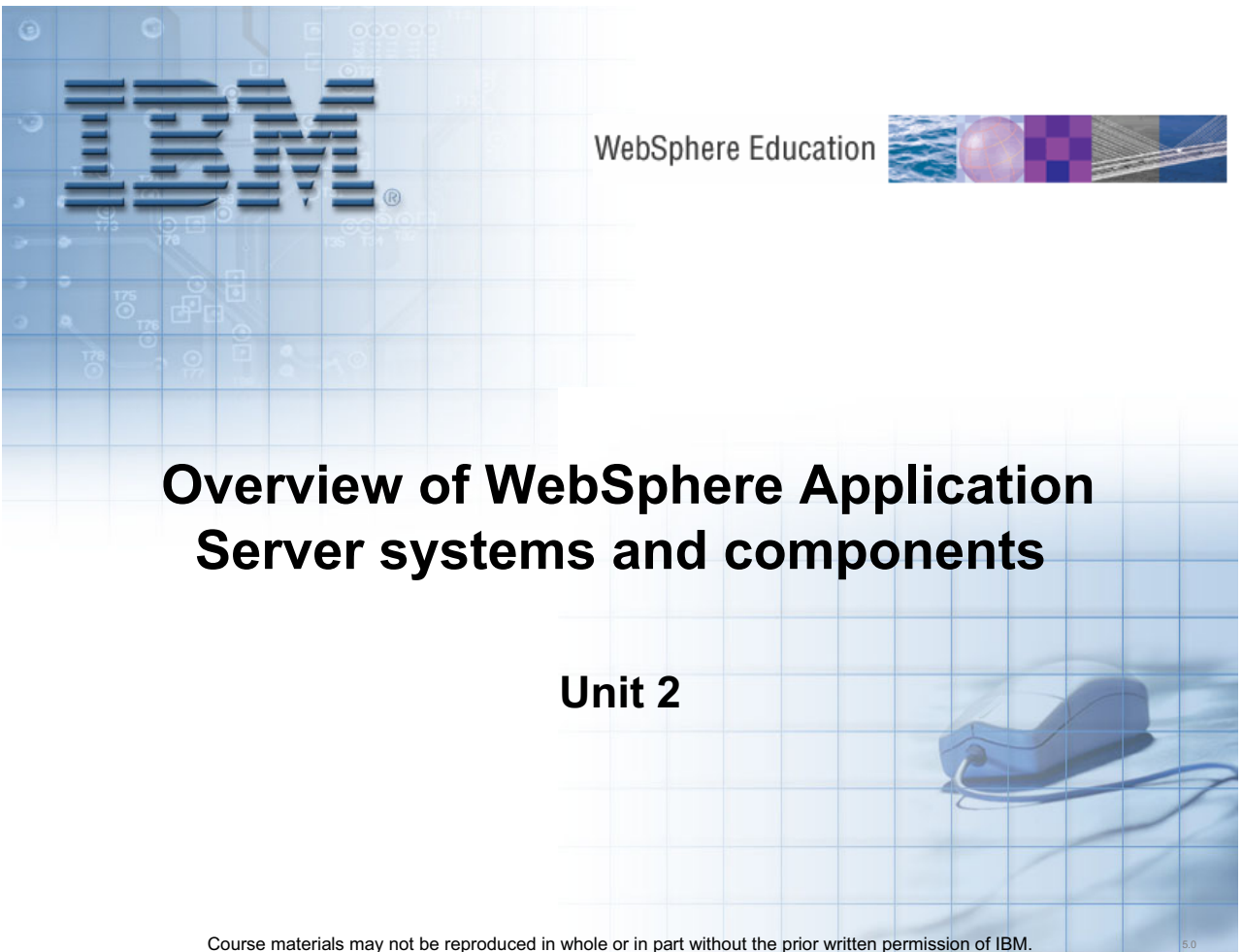
- Describe network deployment cell architecture
- List and describe the function of IBM products involved in implementing stand-alone and distributed architectures
- Identify the components within the application server and describe the services they provide
- Identify the components of an ND Cell and describe the function of each
- Describe stand-alone server architecture
- Describe the different application server clients
- Describe the flow of an application request
- Describe the flow of administration requests
- Identify common troubleshooting points in the end-to-end flow of client request

How you will check your progress

- Checkpoint

References

- SG24-7304 *WebSphere Application Server V6.1: System Management and Configuration*
- WA361 *IBM WebSphere Application Server V6.1 Administration*



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.0

Figure 2-1. Overview of WebSphere Application Server systems and components

WA5711.0

Notes:

Instructor notes:

What students will do —

How students will do it —

What students will learn —

How this will help students on their job —

Unit objectives

After completing this unit, you should be able to:

- Describe stand-alone server architecture
- Describe network deployment cell architecture
- List and describe the function of IBM products involved in implementing stand-alone and distributed architectures
- Identify the components within the WebSphere Application Server and describe the services they provide
- Identify the components of an network deployment cell and describe the function of each
- Describe the different application server clients
- Describe the flow of an application request
- Describe the flow of administration requests
- Identify common troubleshooting points in the end-to-end flow of client request

Figure 2-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere Application Server architecture

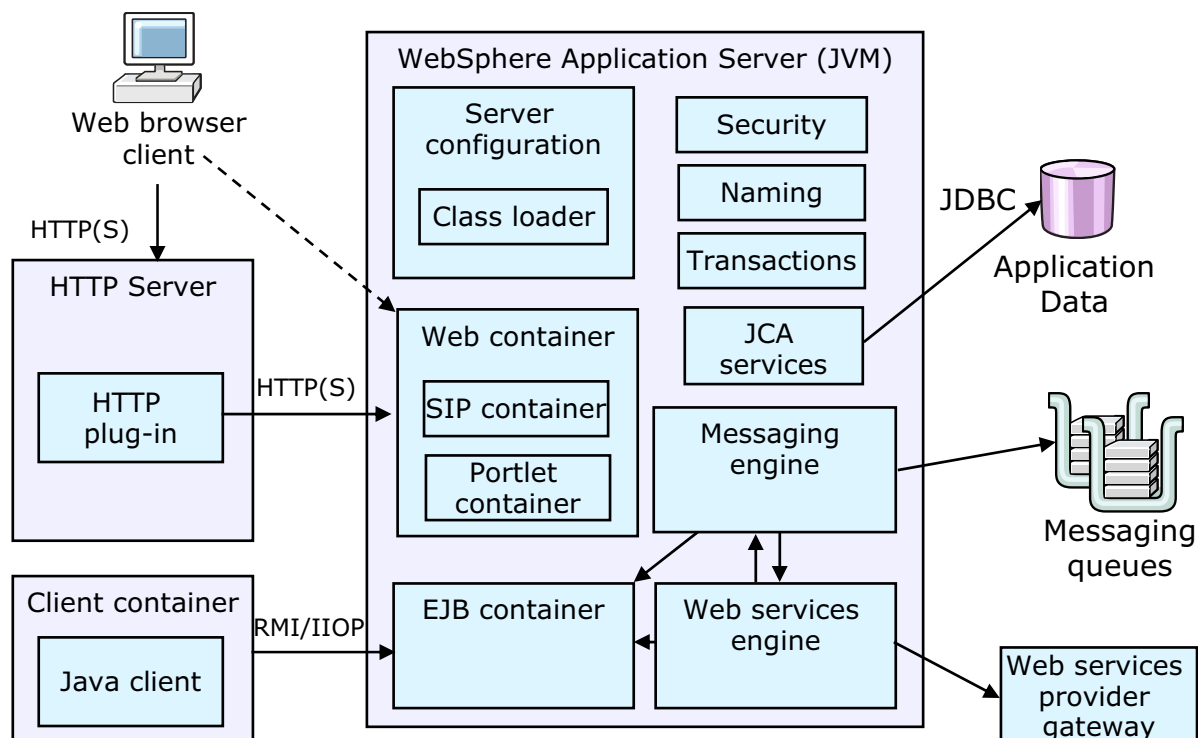


Figure 2-3. WebSphere Application Server architecture

WA5711.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

The main element is the application server, a Java process that encapsulates many services, including the containers, where business logic executes. If you are familiar with J2EE, you will recognize the Web container and the EJB container. The Web container executes servlets and JavaServer Pages (JSPs), both of which are Java classes that generate markup to be viewed by a Web browser. Traffic into and out of the Web container travels through the embedded HTTP Server. While servlets and JSPs can act independently, they most commonly make calls to Enterprise Java Beans (EJBs) to execute business logic or access data. EJBs, which run in the EJB container, are easily reusable Java classes. They most commonly communicate with a relational database or other external source of application data, either returning that data to the Web container or making changes to the data on behalf of the servlet/JSP.

The JMS messaging engine is built into the application server. This is a pure-Java messaging engine. JMS destinations, known as queues and topics provide asynchronous

messaging services to the code running inside the containers. JMS will be covered in more depth later in this course.

As you will see in more detail later on, the Web services engine enables application components to be exposed as Web services, which can be accessed using Simple Object Access Protocol (SOAP).

Several other services run within the application server, including the Dynamic Cache, Data Replication, Security, and others. These will be covered later in the course.

There are also some important components outside of the application server process.

WebSphere Application Server also provides a plug-in for HTTP servers that determines what HTTP traffic is intended to be handled by WebSphere, and routes the requests to the appropriate server. The plug-in is also a critical player in workload management of HTTP requests, as it can distribute the load to multiple application servers, as well as steer traffic away from unavailable servers. It too reads its configuration from a special XML file.

Instructor notes:

Purpose — To show the runtime architecture of a WebSphere Application Server.

Details —

1. The browser is the main interaction mechanism that users will use.
2. A browser communicates with a Web server (HTTP Server).
3. The way the request gets into the WebSphere Application Server is from the HTTP Server Plug-in that is loaded with the HTTP Server. This request is forwarded to the Embedded HTTP server within the application server. The embedded server forwards the request into the Web container to either a servlet or a JSP.
4. If these servlets or JSPs wish to access distributed business logic or a database, the J2EE way to do this is using EJBs within the EJB container.
5. EJBs (entity in this case) can communicate with the database to store, retrieve, query and delete data. JDBC is one way that this communication can occur.
6. There are many other services that are provided within the WebSphere Application Server. Some of those services are listed here.
7. The browser may communicate directly with the embedded HTTP server (bypassing the external Web server) this should only be used for testing and development purposes. This is the way that you will access the application servers in many of the exercises.
8. Browsers are not the only clients, a pure Java client can access EJBs directly using RMI/IIOP.
9. Web services clients can also access the application server. There are essentially two ways that this can happen: 1) using SOAP over HTTP and pass through the embedded HTTP server, and 2) communicating directly to the messaging engine within the application server.
10. Finally, you can have a JMS client that directly communicates with the messaging engine.

Additional information — For steps 4 and 5 above, note that there are many customer applications that access the DB directly from the servlets, without going through EJBs. The non-EJB apps may in fact still be the majority at this point.

Transition statement —

Web container

- The Web container processes
 - Servlets
 - JSP files
 - Other types of server-side includes
- Each application server run time has one logical Web container, which can be modified, but not created or removed
- Each Web container provides the following:
 - Web container transport chains
 - Servlet processing
 - JSP processing
 - HTML and other static content processing
 - Session management
 - Threading support (thread pool)
 - SIP container
 - Portlet container

Figure 2-4. Web container

WA5711.0

Notes:

Web container transport chains

Requests are directed to the Web container using the Web container inbound transport chain. The chain consists of a TCP inbound channel that provides the connection to the network, an HTTP inbound channel that serves HTTP 1.0 and 1.1 requests, and a Web container channel over which requests for servlets and JSPs are sent to the Web container for processing.

Servlet processing

When handling servlets, the Web container creates a request object and a response object, then invokes the servlet service method. The Web container invokes the destroy method of the servlet when appropriate and unloads the servlet, after which the JVM performs garbage collection.

HTML and other static content processing

Requests for HTML and other static content that are directed to the Web container are served by the Web container inbound chain. However, in most cases, using an external Web server and Web server plug-in as a front-end to a Web container is more appropriate for a production environment.

Session management

Support is provided for the `javax.servlet.http.HttpSession` interface as described in the servlet application program interface (API) specification.

Web services engine

Web services are provided as a set of APIs in cooperation with the J2EE applications. Web services engines are provided to support Simple Object Access Protocol (SOAP).

The following properties allow you configure the services provided by the Web container.

Default virtual host

This is the default virtual host to use for applications on the server.

Enable servlet caching

You can use dynamic cache to improve application performance by caching the output of servlets, commands, and JSPs. This setting allows you to enable dynamic caching for servlets. You must first enable dynamic caching and create the appropriate cache policies in order to use servlet caching.

Session Management

You can determine how the Web container will manage HTTP session data. This includes settings for the session tracking mechanism (for example, cookies), session timeout, and for the session persistence method.

Web container transport chains

You can add to or configure the communication channels used for accessing applications in the Web container. By default, you have four transport chains predefined. These are for secure and nonsecure administration console access, and for default access to the Web container. The transport chains are related to port definitions seen communications section. Port numbers must be unique for each application server instance on a given machine.

Custom Properties

You can specify name/value pairs for configuring internal system properties. Some components can make use of custom configuration properties, which can be defined here. It is not common to pass information to the Web container this way, but the J2EE specification indicates this as a requirement. Most configuration information can be handled programmatically, or through the deployment descriptor.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Enterprise JavaBeans (EJB) container

- The Enterprise JavaBeans (EJB) container provides all the runtime services that are needed to deploy and manage enterprise beans.
- It is a server process that handles requests for session beans, entity beans, and message-driven beans (MDBs)
- The enterprise beans, packaged in EJB modules, installed in an application server do not communicate directly with the server.
- Instead, the EJB container provides an interface between the enterprise beans and the server.
- Together, the container and the server provide the enterprise bean runtime environment.
- The container provides many low-level services including
 - Threading support (thread pool)
 - Transaction support
- The container manages data storage and retrieval for the contained EJBs
- A single container can host more than one EJB Java archive (JAR) file.

Figure 2-5. Enterprise JavaBeans (EJB) container

WA5711.0

Notes:

The following properties allow you configure the services provided by the EJB container.

Passivation Directory

This attribute provides the directory that you can use to store the persistent state of passivated, stateful session EJBs. If you are using the EJB container to manage session data, you should give WebSphere the ability to swap data to disk when necessary. This directory tells WebSphere where to hold EJB session data when it passivates and activates beans from the pool.

Inactive pool cleanup interval

Because WebSphere builds a pool of EJBs to satisfy incoming requests, you need to tell it when to remove beans from this pool to preserve resources. This attribute allows you to define the interval at which the container examines the pools of available bean instances to determine if some instances can be deleted to reduce memory usage.

Default data source JNDI name

Here you can set a default data source to use for EJBs that have no individual data source defined. This setting is not applicable for EJB 2.x-compliant CMP beans.

Initial state

This attribute allows you to identify the state of the container when WebSphere is started. If you have to recycle the application server, this attribute is used to determine whether to start the EJB container at server startup. You would only set this to stopped if you planned on never using the EJB container or EJBs within that specific application server instance.

EJB cache settings

You can set up two types of cache settings in WebSphere:

- **Cleanup interval:** This attribute allows you to set the interval at which the container attempts to remove unused items from the cache in order to reduce the total number of items in cache to the value you set in the cache size attribute.
- **Cache size:** This attribute specifies the number of buckets in the active instance list within the EJB container. This attribute is used by WebSphere to determine how large the cache will be and when to remove components from the cache to reduce its size.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere Application Server services

- J2EE Connector Architecture service (JCA)
- Transaction service
- Data Replication Service (DRS)
- Message listener service
- Object Request Broker (ORB) service
- High availability (HA) manager
- Administrative service (JMX)
- Diagnostic trace and Debugging service
- Name service (JNDI)
- Performance Monitoring Interface service
- Security service (JAAS and Java 2 security)
- Session Initiation Protocol (SIP) container
- Service Integration Bus (SIBus) service

Figure 2-6. WebSphere Application Server services

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — Most of these services will be described in detail in the slides that follow.

Additional information —

Transition statement —

J2EE Connector Architecture service (JCA)

- The JCA Connection Manager administers:
 - Connections obtained through resource adapters defined by the JCA
 - Data sources defined by the JDBC 2.0 Extensions

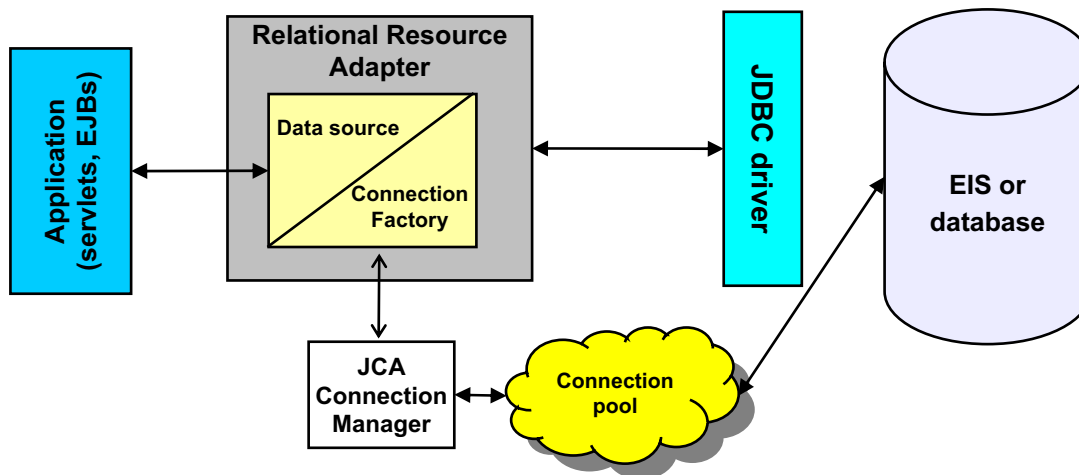


Figure 2-7. J2EE Connector Architecture service (JCA)

WA5711.0

Notes:

How applications use transactions depends on the type of application component, for example:

- A session bean can either use container-managed transactions where the bean delegates management of transactions to the container, or bean-managed transactions where the bean manages transactions itself.
- Entity beans use container-managed transactions.
- Web components, or servlets, use bean-managed transactions.

Instructor notes:

Purpose —

Details — WebSphere Application Server handles transactions with three main components:

- A transaction manager supports the enlistment of recoverable XAResources and ensures that each such resource is driven to a consistent outcome, either at the end of a transaction, or after a failure and restart of the application server.
- A container in which the J2EE application runs manages the enlistment of XAResources on behalf of the application when it performs updates to transactional resource managers such as databases. Optionally, the container can control the demarcation of transactions for enterprise beans that are configured for container-managed transactions.
- A UserTransaction API handles bean-managed enterprise beans and servlets. UserTransaction allows such application components to control the demarcation of their own transactions.

Additional information —

Transition statement —

Transaction service

- WebSphere applications use transactions to coordinate multiple updates to resources as one unit of work.
- Transactions are started and ended by applications or the container. Hence transactions can be configured as either:
 - Container-managed
 - Bean-managed
- WebSphere Application Server is a transaction manager that supports the coordination of resource managers through the XAResource interface and participates in distributed global transactions.
- You can also configure WebSphere applications to interact with:
 - Databases
 - Java Message Service (JMS) queues
 - JCA connectors

Figure 2-8. Transaction service

WA5711.0

Notes:

The dynamic cache works within an application server, intercepting calls to objects that can be cached, for example, through the **service()** method of a servlet or the **execute()** method of a command. The dynamic cache stores the output of the object to the dynamic cache or serves content of the object from the dynamic cache.

Because J2EE applications have high read-write ratios and can tolerate small degrees of latency in the currency of their data, the dynamic cache can create significant gains in server response time, throughput, and scalability.

Cache replication

Cache replication among cluster members takes place using the WebSphere Data Replication Service. Data is generated one time and then copied or replicated to other servers in the cluster, saving execution time and resources.

Cache disk offload

By default, when the number of cache entries reaches the configured limit for a given WebSphere server, eviction of cache entries occurs, allowing new entries to enter the cache service. The dynamic cache includes a disk offload feature that copies the evicted cache entries to disk for potential future access.

Edge Side Include caching

The Web server plug-in contains a built-in Edge Side Include (ESI) processor.

The ESI processor caches whole pages, as well as fragments, providing a higher cache hit ratio. The cache implemented by the ESI processor is an in-memory cache, not a disk cache. Therefore, the cache entries are not saved when the Web server is restarted.

External caching

The dynamic cache controls caches outside of the application server, such as that provided by the Edge components, an IBM HTTP Server FRCA cache, and a WebSphere HTTP Server plug-in ESI Fragment Processor. When external cache groups are defined, the dynamic cache matches external cache entries with those groups and pushes out cache entries and invalidations to those groups. This external caching allows WebSphere to manage dynamic content beyond the application server. The content can then be served from the external cache, instead of the application server, improving performance.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Data Replication Service

- The Data Replication Service (DRS) is responsible for replicating in-memory data among WebSphere processes. You can use DRS for:
 - Stateful session EJB persistence and failover
 - HTTP session persistence and failover
 - Dynamic cache replication
- WebSphere Application Server offers two topologies when setting up data replication among servers:
 - Peer-to-peer topology
 - Client/server topology

Figure 2-9. Data Replication Service

WA5711.0

Notes:

An Object Request Broker (ORB) manages the interaction between clients and servers, using Internet Inter-ORB Protocol (IIOP). The ORB service enables clients to make requests and receive responses from servers in a network-distributed environment. The ORB service provides a framework for clients to locate objects in the network and call operations on those objects as though the remote objects were located in the same running process as the client. The ORB service provides location transparency. The client calls an operation on a local object, known as a *stub*. Then the stub forwards the request to the desired remote object, where the operation is run, and the results are returned to the client. The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request in the network. The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client. The client-side ORB de-marshals the returned results and passes the result to the stub, which returns the result to the client application, as though the operation had been run locally.

WebSphere Application Server uses an ORB to manage communication between client applications and server applications as well as communication among product components.

Instructor notes:

Purpose —

Details — You might mention that the ORB is also involved in communication between EJBs “inside” a single JVM process, although in that case it is used in a rather special mode.

Additional information —

Transition statement —

Name service

- Each application server hosts a name service that provides a Java Naming and Directory Interface (JNDI) namespace.
- Registers all EJB and J2EE resources that are hosted by the application server including:
 - JDBC providers
 - JMS destinations
 - JCA (J2C) components
 - URL providers
 - JavaMail providers
- The deployment manager and all node agents host a name service.
- Configured bindings can map resources to remote locations.

Figure 2-10. Name service

WA5711.0

Notes:

High availability is a very complex topic and will not be covered within this course.

Instructor notes:

Purpose —

Details — As the student notes point out, this is a very complex topic. The intent with this slide is just to make students aware of high availability, not to explore the topic in any depth.

The HA manager runs as a service within each application server process that monitors the healthiness of WebSphere clusters. In the event of a server failure, the HA manager will fail over the singleton service and recover any in-flight transactions. Each application server process runs a HA manager component and shares information through an underlying communication infrastructure Distribution and Consistency Services (DCS) such that no single point of failure would exist in the topology. Every member in a WebSphere cluster would know where singleton services are running.

There is an excellent graphic in the *WebSphere Application Server V6 Scalability and Performance Handbook* redbook (SG24-6392-00).

Additional information —

Transition statement —

Naming topology

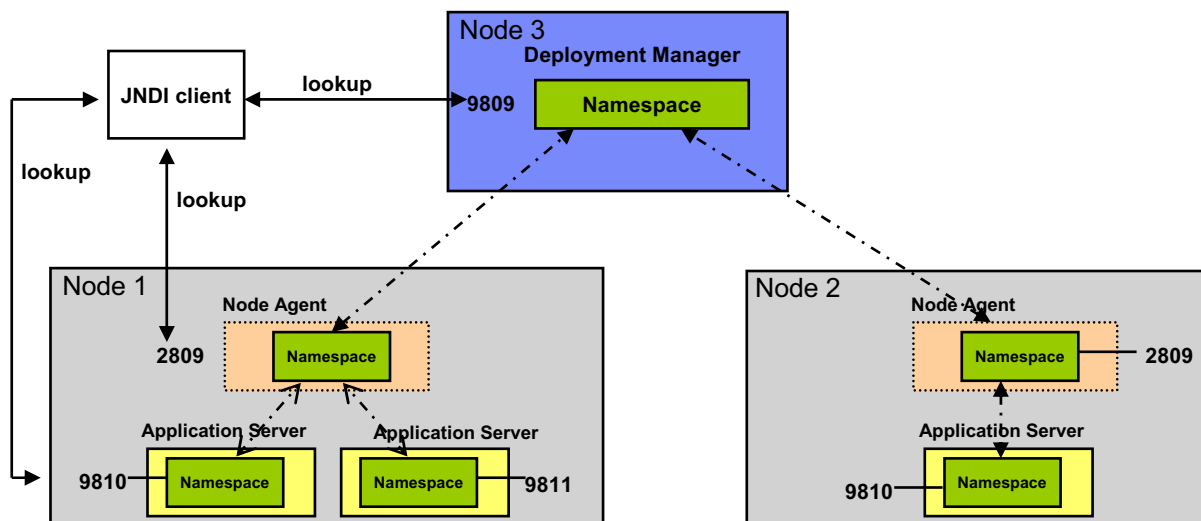


Figure 2-11. Naming topology

WA5711.0

Notes:

The administrative service runs within each server JVM. In Base and Express, the administrative service runs in the application server. In Network Deployment, each of the following hosts an administrative service:

- Deployment manager
- Node agent
- Application server

The administrative service provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository in the server file system.

The administrative service has a security control and filtering functionality that provides different levels of administration to certain users or groups using the following administrative roles:

- Administrator

- Monitor
- Configurator
- Operator

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Object Request Broker service

- An Object Request Broker (ORB) manages the interaction between EJB clients and EJBs, using Internet Inter-ORB Protocol (IIOP).
- The ORB service enables clients to make requests and receive responses from EJBs in a network-distributed environment.
- The ORB service provides a framework for clients to locate objects in the network and call operations on those objects.
- The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request across the network.
- The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client.

Figure 2-12. Object Request Broker service

WA5711.0

Notes:

The service is used to register resources hosted by the application server. The JNDI implementation in WebSphere Application Server is built on top of a Common Object Request Broker Architecture (CORBA) naming service (CosNaming).

JNDI provides the client-side access to naming and presents the programming model that application developers use. CosNaming provides the server-side implementation and is where the namespace is stored. JNDI essentially provides a client-side wrapper of the namespace stored in CosNaming and interacts with the CosNaming server on behalf of the client.

The naming architecture is used by clients of WebSphere applications to obtain references to objects related to those applications. These objects are bound into a mostly hierarchical structure, referred to as a namespace. The namespace structure consists of a set of name bindings, each containing a name relative to a specific context and the object bound with that name. The namespace can be accessed and manipulated through a name server.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

High availability overview

- High availability (HA) manager is used to eliminate single points of failure
- High availability manager is responsible for running key services on available servers rather than on a dedicated one (such as the DMgr)
- Takes advantage of fault-tolerant storage technologies such as Network Attached Storage (NAS)
- Hot standby and peer failover for critical singleton services
 - WLM routing, PMI aggregation, JMS messaging, transaction manager, and so forth
 - Failed singleton starts up on an already-running JVM
 - Planned failover takes < 1 second

Figure 2-13. High availability overview

WA5711.0

Notes:

Distributed namespace

For additional scalability, the namespace for a cell is distributed among the various servers. The deployment manager, node agent, and application server processes all host a name server. The default initial context for a server is its server root. System artifacts, such as EJB homes and resources, are bound to the server root of the server with which they are associated.

Transient and persistent partitions

The namespace is partitioned into *transient* areas and *persistent* areas.

Server roots are transient. System-bound artifacts such as EJB homes and resources are bound under server roots. There is a cell-persistent root that is used for cell-scoped persistent bindings and a node-persistent root that is used to bind objects with a node scope.

Federated namespace structure

A *namespace* is a collection of all names bound to a particular name server. A namespace can contain naming context bindings to contexts located in other servers. If this is the case, the namespace is said to be a *federated* namespace, because it is a collection of name spaces from multiple servers. The namespaces link together to cooperatively form a single logical namespace. In a federated namespace, the real location of each context is transparent to client applications. Clients have no knowledge that multiple name servers are handling resolution requests for a particular requested object.

In a Network Deployment distributed server configuration, the namespace for the cell is federated among the deployment manager, node agents, and application servers of the cell. Each such server hosts a name server. All name servers provide the same logical view of the cell namespace, with the various server roots and persistent partitions of the namespace being interconnected by means of the single logical namespace.

Configured bindings

You can use the configuration graphical interface and script interfaces to configure bindings in various root contexts within the namespace. These bindings are read-only and are bound by the system at server startup.

Support for CORBA Interoperable Naming Service (INS) object Uniform Resource Locator (URL)

WebSphere Application Server contains support for CORBA object URLs (**corbaloc** and **corbaname**) as JNDI provider URLs and lookup names.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative service

- The administrative service runs within each server JVM. In Base and Express, the administrative service runs in the application server.
- In Network Deployment, each of the following hosts an administrative service:
 - Deployment manager
 - Node agents
 - Application servers
 - Managed Web servers
- Provides the necessary functions to manipulate configuration data for the server and its components.
- The configuration is stored in a repository in the server file system.
 - The repository consists of sub-directories which contain XML files of configuration information
- The administrative service is invoked using JMX calls that go through Web services (SOAP/HTTP or RMI/IIOP)

Figure 2-14. Administrative service

WA5711.0

Notes:

Replication domains, consisting of server or cluster members that have a need to share internal data, perform the replication. Multiple domains can be used, each for a specific task among a set of servers or clusters. While HTTP session replication and EJB state replication can (and should) share a domain, you need a separate domain for dynamic cache replication. You can define a domain so that each domain member has a single replicator that sends data to another domain member. You can also define a domain so that each member has multiple replicators that send data to multiple domain members.

WebSphere Application Server offers two topologies when setting up data replication among servers: Peer-to-peer topology and client/server topology.

Peer-to-peer topology

Each application server stores sessions in its own memory and retrieves sessions from other application servers. In other words, each application server acts as a *client* by retrieving sessions from other application servers. Each application server also acts as a

server by providing sessions to other application servers. This mode, working in conjunction with the workload manager, provides hot failover capabilities.

Client/server topology

Client application servers send session information to the replication servers and retrieve sessions from the servers. They respond to user requests and store only the sessions of the users with whom they interact. Application servers act as either a replication client or a server. Those that act as replication servers store sessions in their own memory and provide session information to clients. They are dedicated replication servers that store sessions but do not respond to user requests.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SIBus Web services

- A software component that enables your Web services to use service integration technologies
- Provides a choice of quality of service and message distribution options for Web services
- Provides intelligence in the form of mediations, which allow for the rerouting of messages

Figure 2-15. SIBus Web services

WA5711.0

Notes:

Connection management for access to enterprise information systems (EIS) in WebSphere Application Server is based on the J2EE Connector Architecture (JCA) specification, also sometimes referred to as J2C. The connection between the enterprise application and the EIS is done through the use of EIS-provided resource adapters, which are plugged into the application server. The architecture specifies the connection management, transaction management, and security contracts that exist between the application server and the EIS.

Within the application server, the Connection Manager pools and manages connections. The Connection Manager administers connections that are obtained through both resource adapters defined by the JCA specification and data sources defined by the JDBC 2.0 Extensions, and later, specification.

The JCA Connection Manager provides the connection pooling, local transaction, and security supports. The relational resource adapter provides the JDBC wrappers and JCA CCI implementation that allow applications using bean-managed persistence, JDBC calls, and container-managed persistence beans to access the database JDBC driver.

The JCA resource adapter is a system-level software driver supplied by EIS vendors or other third-party vendors. It provides the connectivity between J2EE components (an application server or an application client) and an EIS.

One resource adapter, the WebSphere Relational Resource Adapter, is predefined for handling data access to relational databases. This resource adapter provides data access through JDBC calls to access databases dynamically. It provides connection pooling, local transaction, and security support. The WebSphere persistence manager uses this adapter to access data for container-managed persistence beans.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Security service

- Each application server JVM hosts a Security service.
- The Security service uses the security settings held in the configuration repository to provide authentication and authorization functionality.
- User registries containing authentication data can be configured using one of the following:
 - Local OS
 - LDAP
 - Custom user registry
 - Federated repository

Figure 2-16. Security service

WA5711.0

Notes:

To trace the SIP container, replace the content of the trace specification with the following code: `com.ibm.ws.sip.*=all=enabled`. Tracing will be covered in more detail later in the course.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Additional services

- Message listener service
 - Uses listener ports to support EJB 2.0 message-driven beans
- Administrative infrastructure
 - Used to manage cell configuration and environment
- Performance Monitoring Infrastructure (PMI) service:
 - Used to collect data on the server runtime components and application components

Figure 2-17. Additional services

WA5711.0

Notes:

If you encounter a problem that you think might be related to SIBus Web services, you can check for error messages in the WebSphere Application Server administrative console, and in the application server SystemOut.log file. You can also enable the application server debug trace to provide a detailed exception dump.

Viewing logs and messages, and enabling tracing will be covered in detail later in the course.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere runtime flow

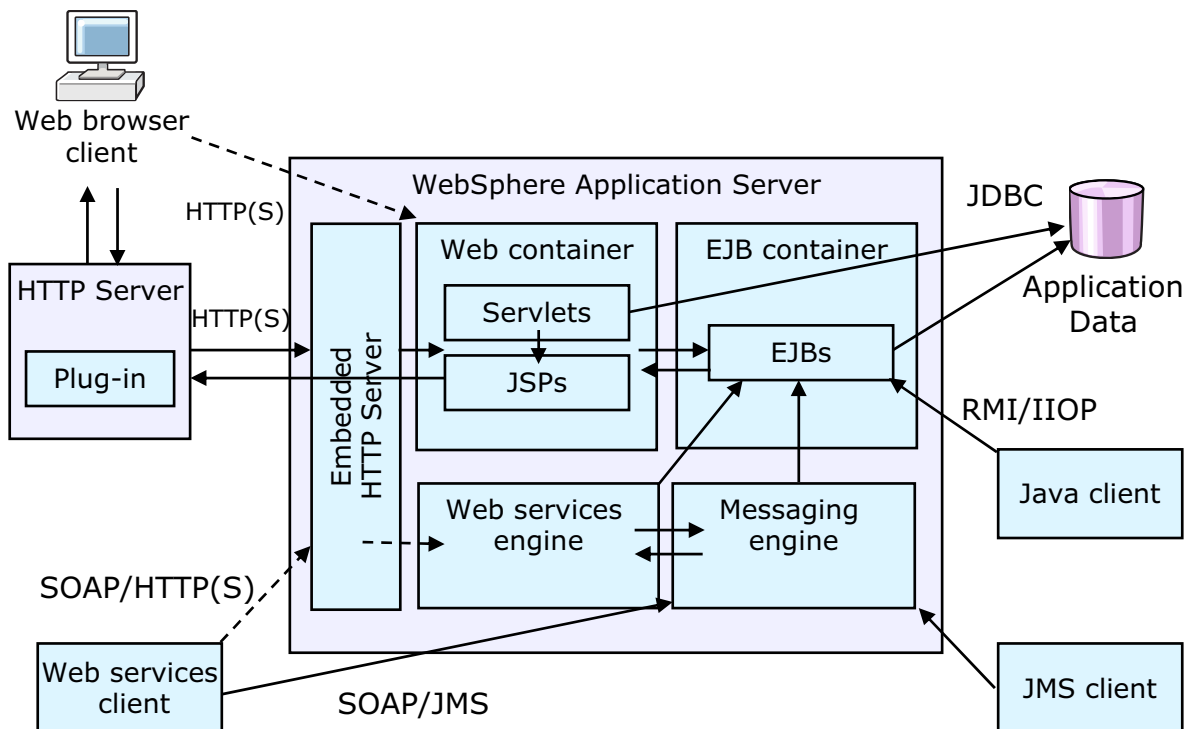


Figure 2-18. WebSphere runtime flow

WA5711.0

Notes:

Message listener service

With EJB 2.1, an `ActivationSpec` is used to connect message-driven beans to destinations. However, you can deploy existing EJB 2.0 message-driven beans against a listener port as in WebSphere Application Server V5. For those message-driven beans, the Message listener service provides a listener manager that controls and monitors one or more JMS listeners. Each listener monitors a JMS destination on behalf of a deployed message-driven bean.

Performance Monitoring Infrastructure (PMI)

PMI is used by WebSphere Application Server to collect data on run time and applications. This infrastructure is compatible with and extends the JSR-077 specification. PMI uses a client/server architecture. The server collects performance data from various WebSphere Application Server components and stores it in memory.

This data consists of counters such as servlet response time and data connection pool usage. The data can then be retrieved using a Web client, Java client, or Java Management Extensions (JMX) client. WebSphere Application Server contains Tivoli Performance Viewer, which is integrated into the WebSphere administrative console and displays and monitors performance data. WebSphere Application Server also collects data by timing requests as they travel through the product components. PMI request metrics log the time spent in major components, such as Web containers, EJB containers, and databases. These data points are recorded in logs and can be written to Application Response Time agents that Tivoli monitoring tools use.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Network deployment concepts

- A node is a logical grouping of application servers.
 - Each node is managed by a single node agent process.
 - Multiple nodes can exist on a single machine through the use of profiles.
- A deployment manager (DMgr) process manages the node agents.
 - Holds the configuration repository for the entire management domain, called a cell.
 - Within a cell, the administrative console runs inside the DMgr.

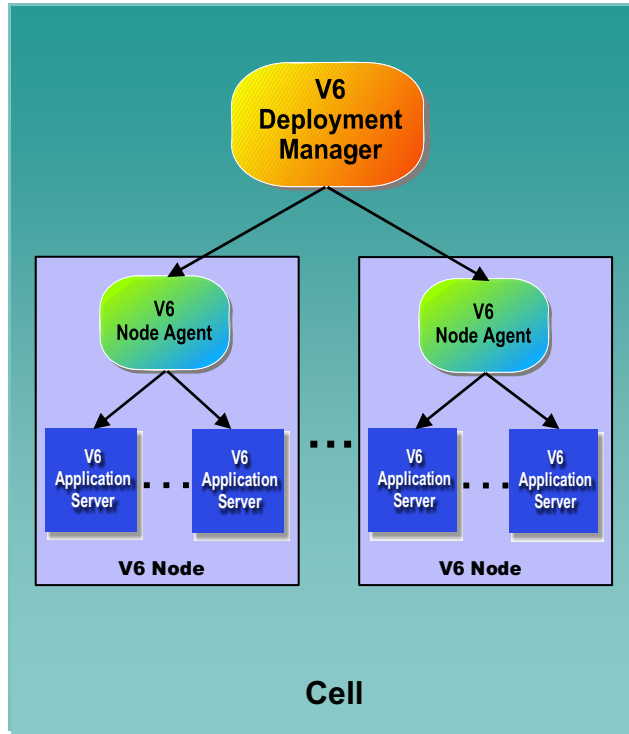


Figure 2-19. Network deployment concepts

WA5711.0

Notes:

The deployment manager here is an application server that manages the administrative environment within a cell. You will see later in this unit that a node is represented by a profile. The node agent is a very important process that allows for communication of administrative information (commands and configuration files) to reach the applications servers.

Instructor notes:**Purpose —**

Details — Spend some time explaining the terms here. Make sure that you point out that the main concepts here are administrative in purpose only, not run time.

**Important**

Also make sure students understand the fact that node does not equal machine.

Additional information —**Transition statement —**

Network deployment runtime flow

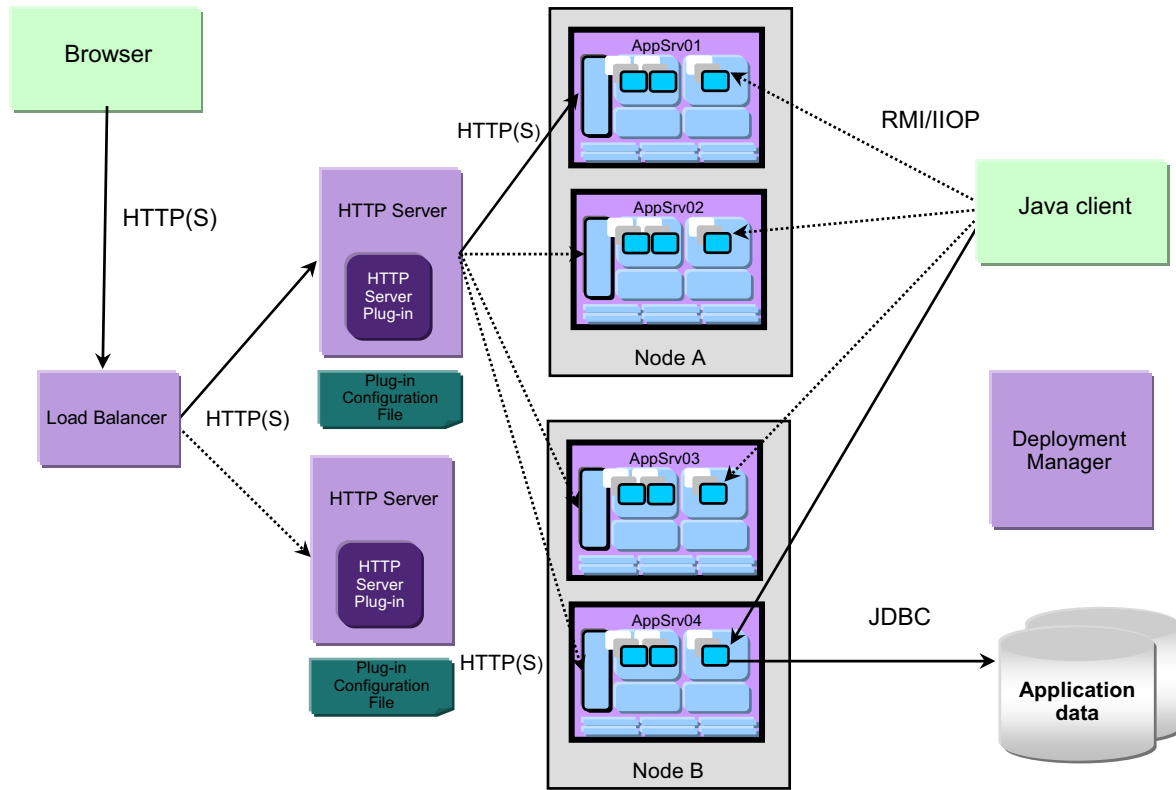


Figure 2-20. Network deployment runtime flow

WA5711.0

Notes:

The main theme with Network Deployment is distributed applications. While the “flow” of an application remains the same, there are significant additions to run time of an application. Note the Load Balancer allows for multiple HTTP servers; users point their browsers to the Load Balancer and their request will be work load-managed to an HTTP Server. Once the request hits one of these HTTP Servers, the HTTP Server Plug-in will load balance the request between the application servers that it is configured to serve. Once the request enters the application server, the flow is identical to how it was in Express and Base.

The Java clients requests to EJBs can also be work load-managed so that the requests do not all hit one application server.

Instructor notes:

Purpose —

Details — You should walk through it with the same process and point out where workload management occurs.

Additional information —

Transition statement —

Network deployment administration flow

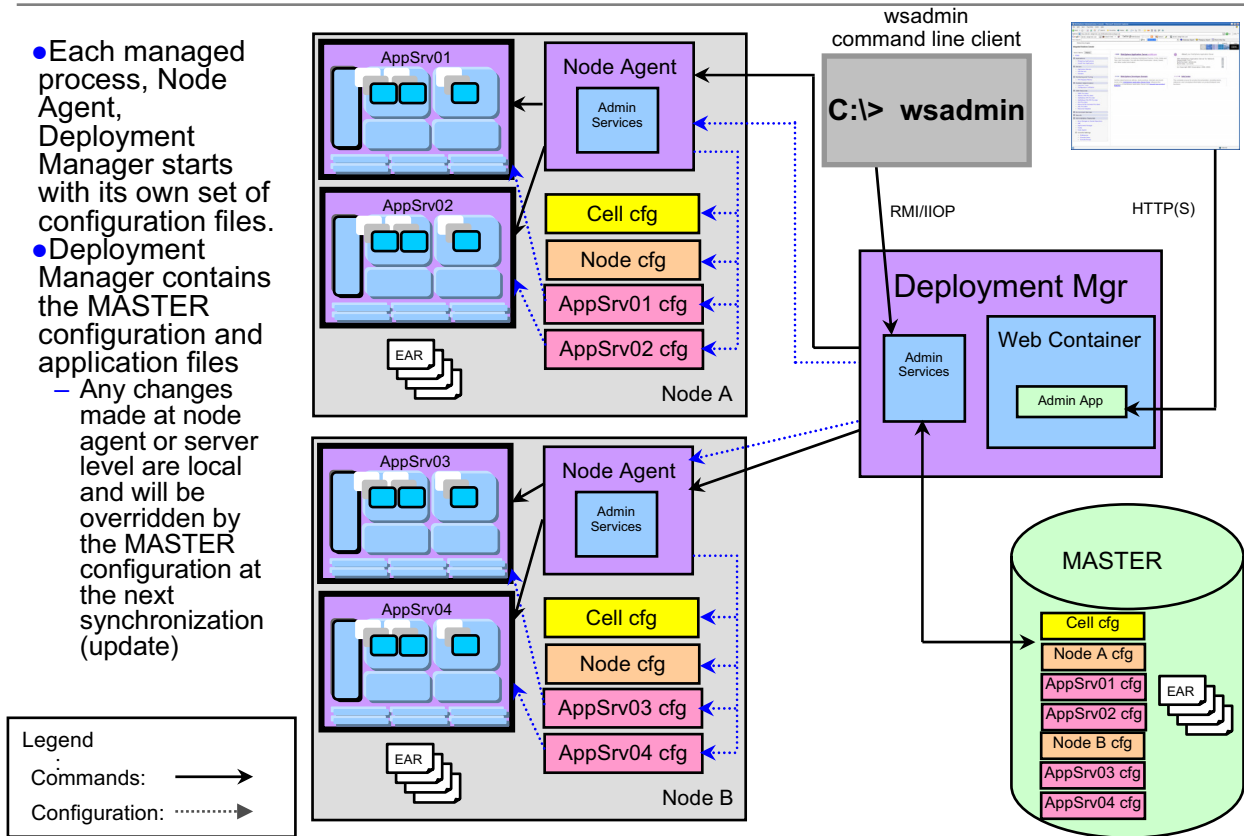


Figure 2-21. Network deployment administration flow

WA5711.0

Notes:

The administrative console and wsadmin are still the two ways that the environment is administered. However, take note that these tools now communicate to the deployment manager and not to the application servers directly. The communication of these commands flows from the tools to the deployment manager to the node agents, to the application servers. This allows administration of multiple nodes (each possibly containing multiple application servers) from a single focal point (the deployment manager).

There is one main repository for the configuration files within a cell, and those are associated with the deployment manager. All updates to the configuration files should go through the deployment manager. You will see in a moment how this process works. You should be very careful in connecting to an application server directly with wsadmin or the administrative console as any changes that are made to the configuration files are only temporary, they will be overwritten with the configuration files from the MASTER files.

Instructor notes:**Purpose —**

Details — It may help to explain up-front that there are really 3 distinct administration flows:

- The flow when an administrator issues a command through wsadmin or the console
- The flow for synchronizing configuration files from the DMGR to the node agents
- The flow used by DMGR to discover node agents when they come up, and by node agents to discover application servers

Additional information —**Transition statement —**

File synchronization and file transfer

- Deployment Manager contains the “master” configuration
- Node agents synchronize their files with the “master” copy
 - Automatically
 - At start up
 - Periodically
 - Manually
 - Administrative console
 - Command line
- During synchronization
 - Node agent checks for changes to master configuration
 - New or updated files are copied to the node

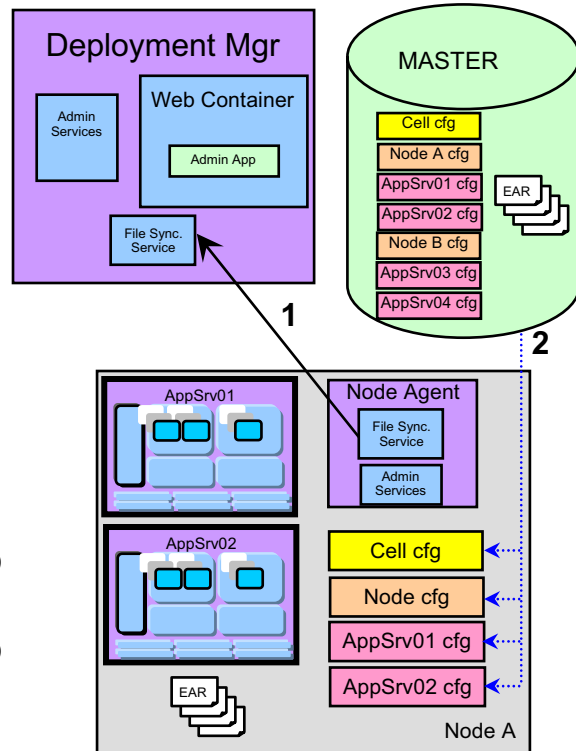


Figure 2-22. File synchronization and file transfer

WA5711.0

Notes:

File synchronization service

The administrative service responsible for keeping up to date the configuration and application data files that are distributed across the cell. The service runs in the deployment manager and node agents, and ensures that changes made to the master repository will be propagated out to the nodes, as necessary. The file transfer system application is used for the synchronization process. File synchronization can be forced from an administration client, or can be scheduled to happen automatically. During the synchronization operation, the node agent checks with the deployment manager to see if any files that apply to the node have been updated in the master repository. New or updated files are sent to the node, while any deleted files are also deleted from the node. Synchronization is one-way. The changes are sent from the deployment manager to the node agent. No changes are sent from the node agent back to the deployment manager.

Instructor notes:

Purpose —

Details — Once again reinforce that this process will overwrite changes made directly on the node.

Additional information —

Transition statement —

Web servers

- Web servers can be defined to the administration process as Web server nodes, allowing applications to be associated with one or more defined Web servers.
- Web server nodes can be *managed* or *unmanaged*.
- *Managed* nodes have a node agent on the Web server machine that allows the deployment manager to administer the Web server. You can:
 - Start or stop the Web server from the deployment manager
 - Generate the Web server plug-in configuration file for the node
 - Automatically push the file to the Web server
- Managed Web server nodes are usually behind the firewall with WebSphere Application Server installations.
- *Unmanaged* Web server nodes, as the name implies, are not managed by WebSphere.
 - Normally found outside the firewall, or in the demilitarized zone.
 - You must manually copy or FTP Web server plug-in configuration files to the Web server.
 - However, if you define the Web server as a node, you can generate custom plug-in configuration files for it.

Figure 2-23. Web servers

WA5711.0

Notes:

As a special case, if the unmanaged Web server is an IBM HTTP Server, you can administer the Web server from the WebSphere administrative console. Then, you can automatically push the plug-in configuration file to the Web server with the deployment manager using HTTP commands to the IBM HTTP Server administration process. This configuration does not require a node agent.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web server plug-ins

- Web containers can serve static content.
 - But more likely an external Web server receives client requests
- Web servers usually serve requests that do not require dynamic content such as HTML pages.
- When a request requires dynamic content, such as JSP or servlet processing, it must be forwarded to the application server.
- The Web server plug-in forwards a request based on its URI.
- The Web server plug-in is also responsible for load balancing and failover among server cluster members using `plugin-cfg.xml` files.

Figure 2-24. Web server plug-ins

WA5711.0

Notes:

To forward a request, you use a Web server plug-in that is included with the WebSphere Application Server packages for installation on a Web server. You copy an Extensible Markup Language (XML) configuration file, located on the WebSphere Application Server, to the Web server plug-in directory. The plug-in uses the configuration file to determine whether a request should be handled by the Web server or an application server. When WebSphere Application Server receives a request for an application server, it forwards the request to the appropriate Web container in the application server. The plug-in can use HTTP or HTTPS to transmit the request.

Instructor notes:

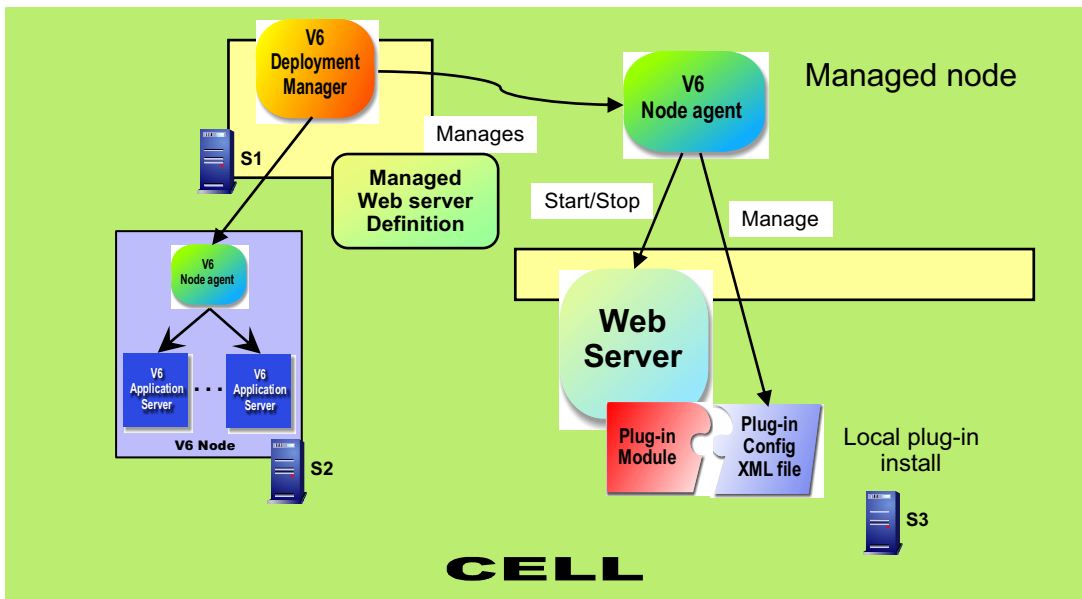
Purpose —

Details —

Additional information —

Transition statement —

Web server: Managed node (local)



- Install Web server on a managed node
- Create a Web server definition within the DMgr
- Node agent receives commands from DMgr to administer the Web server
- Plugin-cfg.xml file is propagated through the file synchronization service and is located in the config directory.

Figure 2-25. Web server: Managed node (local)

WA5711.0

Notes:

The Web server is managed using the deployment manager through the node agent. This provides ability to start, stop the Web server and automatically push the plug-in configuration file to the Web server from the DMgr. This can be used when the Web server is on the same machine where WebSphere Application Server is installed. This is a common scenario behind a firewall where a WebSphere node can be installed.

Instructor notes:

Purpose —

Details — The details are in the student notes. Make sure that you emphasize that here that there is a node agent that is performing the communication with the Web server from the administrative tools of WebSphere.

Additional information —

Transition statement —

Typical client request application flow (1 of 2)

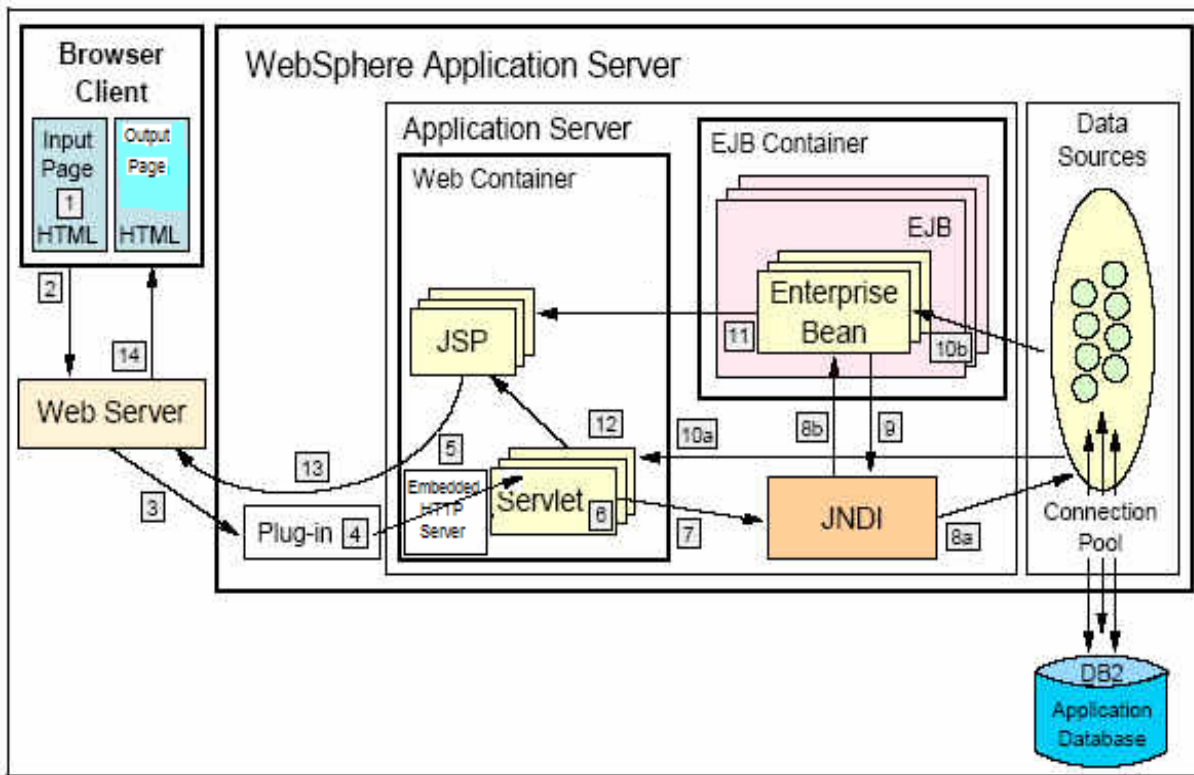


Figure 2-26. Typical client request application flow (1 of 2)

WA5711.0

Notes:

The typical application flow is as follows:

1. A Web client requests a URL in the browser input page.
2. The request is routed to the Web server over the Internet.
3. The Web server immediately passes the request to the Web server plug-in. All requests go to the WebSphere plug-in first.
4. The Web server plug-in examines the URL, verifies the list of host name aliases from which it will accept traffic based on the virtual host information, and chooses a server to handle the request.
5. A stream is created. A *stream* is a connection to the Web container. It is possible to maintain a stream over a number of requests. The Web container receives the request and, based on the URL, dispatches it to the proper servlet.
6. If the servlet class is not loaded, the dynamic class loader loads the servlet (servlet **init()**, then **doGet()** or **doPost()**).

7. JNDI is used for lookup of either data sources or EJBs required by the servlet.
8. Depending upon whether a data source is specified or an EJB is requested, the JNDI directs the servlet:
 - To the corresponding database and gets a connection from its connection pool in the case of a data source.
 - To the corresponding EJB container, which then instantiates the EJB when an EJB is requested.
9. If the EJB requested involves an SQL transaction, it goes back to the JNDI to look up the data source.
10. The SQL statement is executed and the data retrieved is sent back either to the servlet or to the EJB.
11. Data beans are created and handed off to JSPs in the case of EJBs.
12. The servlet sends data to JSPs.
13. The JSP generates the HTML that is sent back through the WebSphere plug-in to the Web server.
14. The Web server sends the output HTML page to the browser.

Instructor notes:

Purpose —

Details — Important to note that this is just a simple example. There are many variations and additional subtleties that can come into play. For example:

- A direct invocation from Web browser to JSP (no servlet)
- Transactions
- Security
- Dynacache

Additional information —

Transition statement —

Typical client request application flow (2 of 2)

1. A Web client requests a URL in the browser input page.
2. The request is routed to the Web server over the Internet.
3. Web server passes the request directly to the Web server plug-in.
4. The Web server plug-in examines the URL, verifies the list of host name aliases from which it will accept traffic based on the virtual host information, and chooses a server to handle the request.
5. A stream is created. A stream is a connection to the Web container. The Web container receives the request and, based on the URL, dispatches it to the proper servlet.
6. If the servlet class is not loaded, the dynamic class loader loads the servlet.
7. JNDI is used for lookup of either data sources or EJBs required by the servlet.
8. Depending upon whether a data source is specified or an EJB is requested, the JNDI directs the servlet to the:
 - a. Database and gets a connection from its connection pool
 - b. EJB container, which then instantiates the EJB when an EJB is requested.
9. If the EJB requested involves an SQL transaction, it uses JNDI to look up the data source.
10. The SQL statement is executed and the data retrieved is sent back either to the servlet or to the EJB.
11. Data beans are created and handed off to JSPs in the case of EJBs.
12. The servlet sends data to JSPs.
13. The JSP generates the HTML that is sent back through the WebSphere plug-in to the Web server.
14. The Web server sends the output HTML page to the browser.

Figure 2-27. Typical client request application flow (2 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Edge Components

- WebSphere Application Server Network Deployment package contains the following Edge Component functionality:
 - Load Balancer
 - Caching proxy
- Edge Components install separately from WebSphere Application Server.
- Load Balancer is responsible for balancing the load across multiple servers that can be within either local area network or wide area network.
- The purpose of the caching proxy is to reduce network congestion within an enterprise by offloading security and content delivery from Web servers and Application Servers.

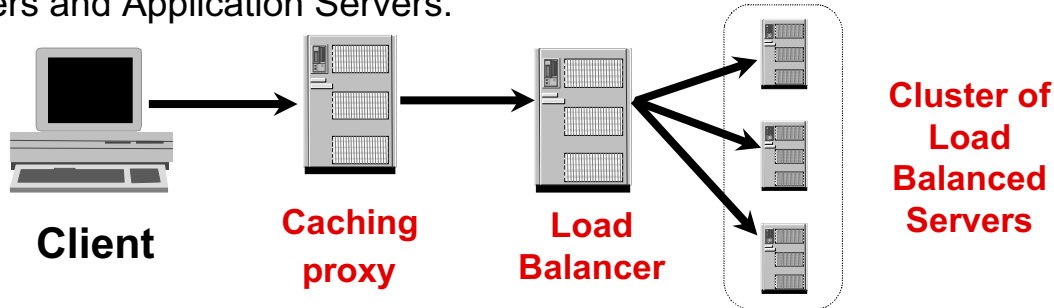


Figure 2-28. Edge Components

WA5711.0

Notes:

The caching proxy intercepts data requests from a client, retrieves the requested information from the application servers, and delivers that content back to the client. It stores cache-able content in a local cache before delivering it to the client. Subsequent requests for the same content are served from the local cache, which is much faster and reduces the network and application server load.

The Load Balancer provides horizontal scalability by dispatching HTTP requests among several, identically configured Web server or application server nodes.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

All components in application flow accessible

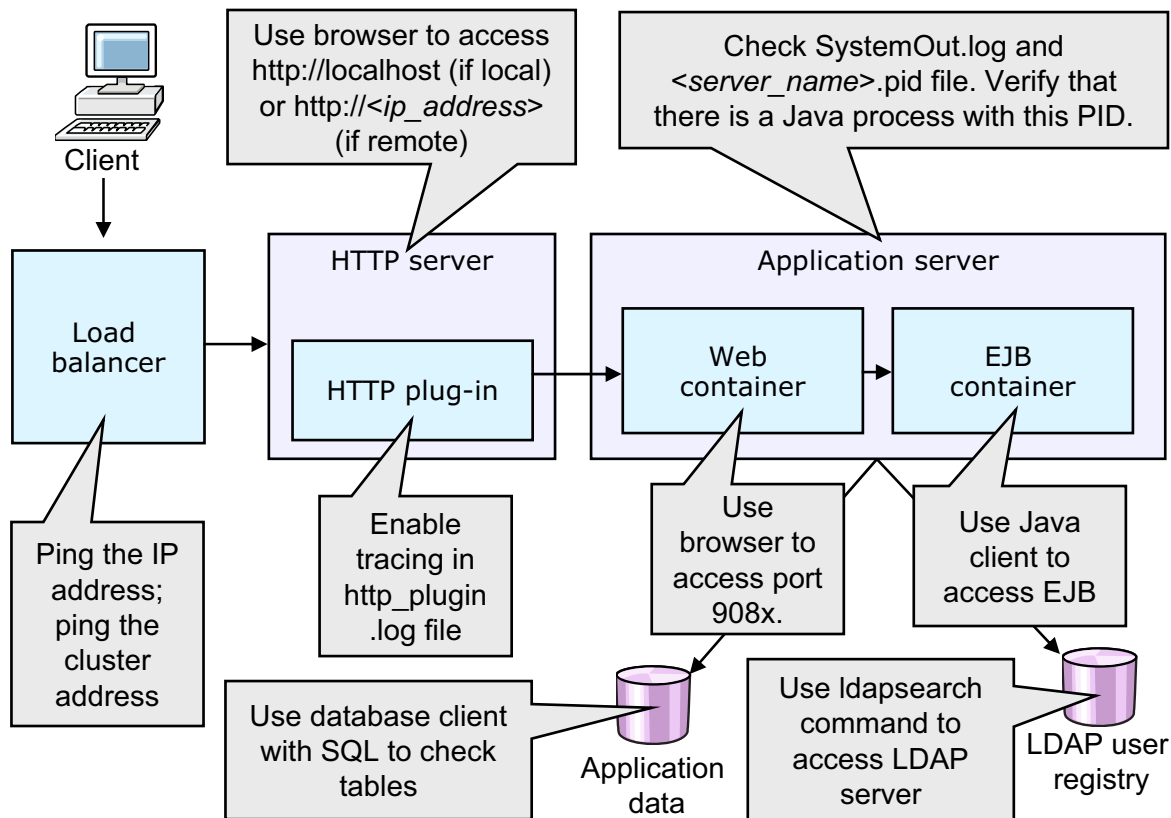


Figure 2-29. All components in application flow accessible

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — Use this slide to initiate a discussion of end-to-end access verification and problem isolation.

Additional information —

Transition statement —

Checkpoint (1 of 2)

1. Which of the following provides an environment for running servlets?
 - a. Web container
 - b. EJB container
 - c. The dynamic cache

2. Which of the following components are contained within the JVM of the application server? (Choose two)
 - a. Messaging engine
 - b. Embedded HTTP Server
 - c. HTTP Server plug-in
 - d. WebSphere MQ Provider
 - e. Cloudscape database
 - f. DB2 database

Figure 2-30. Checkpoint (1 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — Have student circle the correct answers.

Additional information —

Transition statement —

Checkpoint (2 of 2)

3. What protocol does an external Web server use to communicate with the application server?
 - a. JDBC
 - b. SOAP
 - c. HTTP

4. Which of the following is responsible for restarting servers?
 - a. Deployment manager
 - b. Node agent
 - c. Admin service

5. The name service resolves references to all of the resources except which of the following?
 - a. Data sources
 - b. JMS destinations
 - c. Generic servers

Figure 2-31. Checkpoint (2 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — Checkpoint solutions

1. (a) Web container
2. (a) Messaging engine, and (b) Embedded HTTP server
3. (c) HTTP
4. (b) Node agent
5. (c) Generic servers

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe stand-alone server architecture
- Describe network deployment cell architecture
- List and describe the function of IBM products involved in implementing stand-alone and distributed architectures
- Identify the components within the WebSphere Application Server and describe the services they provide
- Identify the components of an network deployment cell and describe the function of each
- Describe the different application server clients
- Describe the flow of an application request
- Describe the flow of administration requests
- Identify common troubleshooting points in the end-to-end flow of client request

Figure 2-32. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 3. Using the IBM Support Assistant

Estimated time

00:30

What this unit is about

This module explains how to use the IBM Support Assistant tool.

What you should be able to do

After completing this unit, you should be able to:

- Describe the different support components such as Search, Product Information, Service, and Tools
- Recognize when to use the IBM Support Assistant
- Update IBM Support Assistant with new software plug-ins
- Install new support tools
- Use Collect Data and Automated PD features
- Use the IBM Support Assistant for problem determination

How you will check your progress

Accountability:

- Machine exercises

References

IBM Support Assistant User Guide:

- <http://www.ibm.com/software/support/isa>

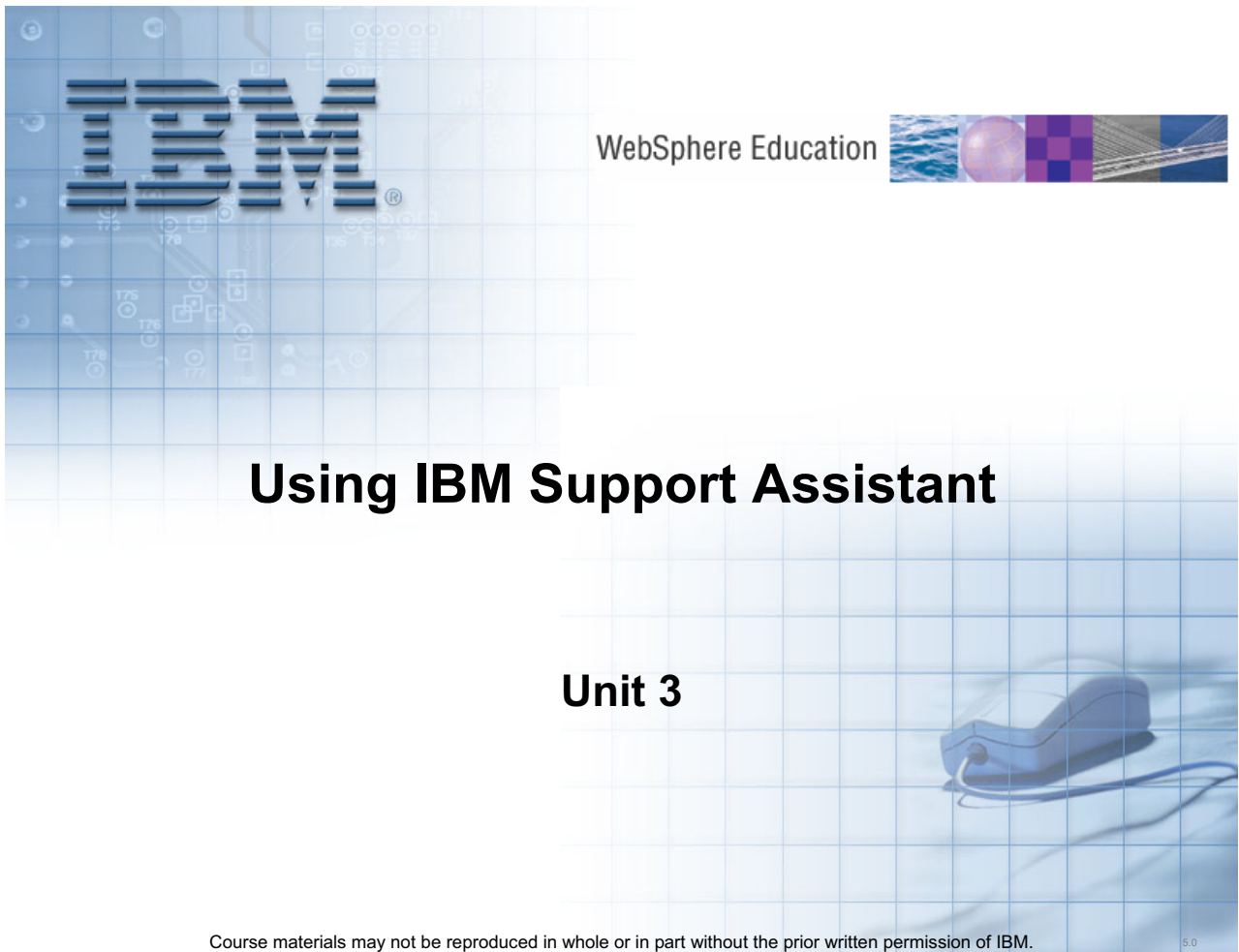


Figure 3-1. Using IBM Support Assistant

WA5711.0

Notes:

Instructor notes:

What students will do —

How students will do it —

What students will learn —

How this will help students on their job —

Unit objectives

After completing this unit, you should be able to:

- Describe the different support components such a Search, Product Information, Service, and Tools
- Recognize when to use the IBM Support Assistant
- Update IBM Support Assistant with new software plug-ins
- Install new support tools
- Use Collect Data and Automated Problem Determination (PD) features
- Use the IBM Support Assistant for problem determination

Figure 3-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — Go over these slides very lightly. Students will benefit most from working with ISA in the following exercise.

Additional information —

Transition statement —

IBM Support Assistant

- The IBM Support Assistant (ISA) is a *free, stand-alone workbench* that you can install on any workstation
- It can be enhanced by installing plug-in modules for the IBM products you use
- Saves time searching product, support, and educational resources
- If a Problem Management Report (PMR) needs to be opened, ISA helps with:
 - Gathering support information based on problem type
 - Creating and updating the problem report
 - Tracking your electronic problem report
- Includes a support tool framework allowing for easy location and installation of useful product support tools
- ISA can be downloaded from <http://www.ibm.com/software/support/isa/>

Figure 3-3. IBM Support Assistant

WA5711.0

Notes:

Using the IBM Support Assistant improves your ability to locate IBM Support, development and educational information through a federated search interface (one search, multiple resources), provides quick access to the IBM Education Assistant and key product education roadmaps, and simplifies access to IBM product home pages, product support pages, as well as product forums or newsgroups, through convenient links. In addition, problems can be submitted to IBM Support by collecting key information, then electronically creating a Problem Management Report (PMR) from within IBM Support Assistant.

The IBM Support Assistant includes a support tool framework allowing for the easy installation of support tools associated with different IBM products and a framework for IBM software products to deliver customized self-help information into the different tools within it. The workbench can be customized through the built-in Updater feature to include the product plug-ins and tools specific to the environment.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM Support Assistant console

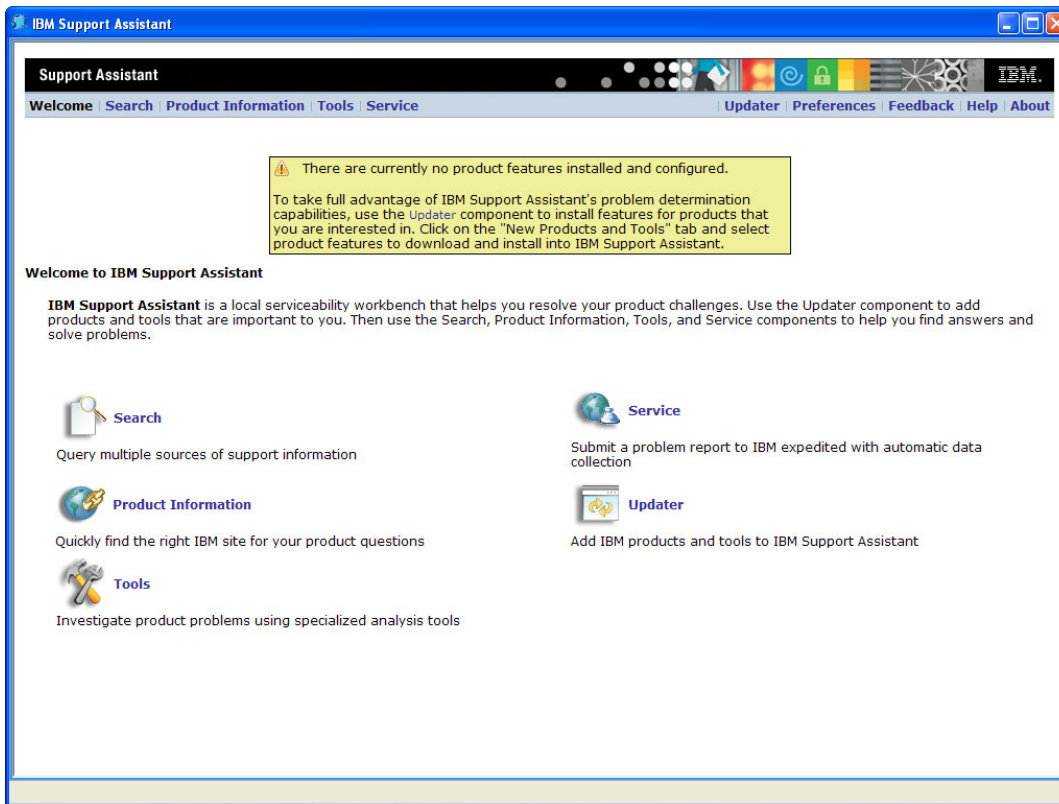


Figure 3-4. IBM Support Assistant console

WA5711.0

Notes:

Download and install ISA

1. Go to <http://www.ibm.com/software/support/isa/>, choose IBM Support Assistant V3, and download the installation package for any platform where you will install IBM Support Assistant.
 - You will need to log in using your IBM Web identity (same ID as used for MySupport, developerWorks, and so forth).
 - If you do not already have an IBM Web identity, you may complete the free registration process to obtain one.
2. Download the compressed archive files.
3. Extract the archive file in a temporary directory using an archive extraction tool.

The archive contains an installer program and the Installation and Troubleshooting Guide. Follow the directions in the Installation and Troubleshooting Guide to install IBM Support Assistant.

Instructor notes:**Purpose —**

Details — The newest version of ISA (v3.1) has an improved updater. However, on slow network connections it is recommended to download one plug-in at a time. In the most severe case, while teaching this class, it might be necessary to use a local update site to perform the labs. Contact the ISA team to request such a site.

Additional information —**Transition statement —**

Updater component

- Product plug-ins and tools can be asynchronously added to the IBM Support Assistant workbench
 - The product plug-ins are organized by brand, such as WebSphere and Rational
 - The exception to this is the *System Z* and *Other* categories
 - The tools are under the *Common Component Tools* branch
- Each category can be expanded
 - Multiple plug-ins are tools can be added simultaneously

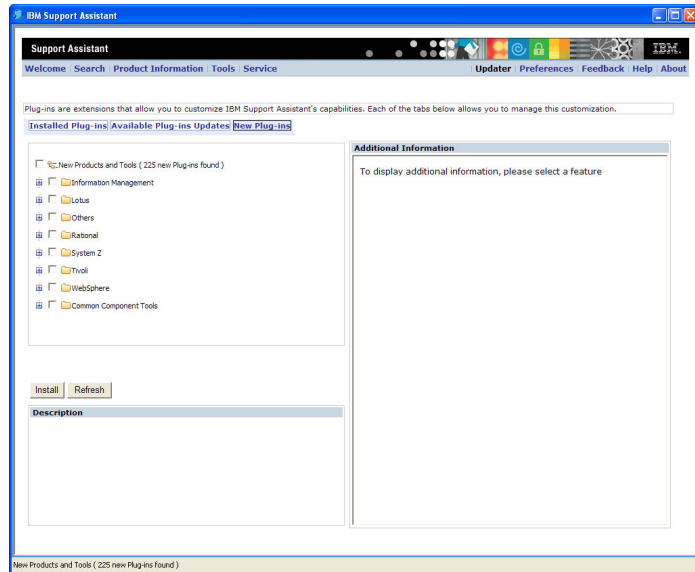
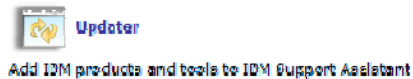


Figure 3-5. Updater component

WA5711.0

Notes:

When you should use the Updater component

The Updater component can be used to locate a more up-to-date version of the product information you currently have installed. If you would like to see more products, you can add them by using the updater to find, download, and install the required product information from the IBM Web site. IBM Support Assistant product information is packaged as product plug-ins. You can add a product by installing a new product plug-in. IBM Support Assistant pluggable tools are also installed by using the updater.

After you install new product information or a pluggable tool, the IBM Support Assistant should restart automatically to pick up the new functionality. If not, you will need to restart IBM Support Assistant to make that information or tool available.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Installing the plug-ins

- The Updater can be used to install available plug-ins for IBM products
- To install the WebSphere Application Server V6.1 plug-in:
 1. Select **Updater -> New Plug-ins -> WebSphere**
 2. Select **WebSphere Application Server V6.1** and click **Install**

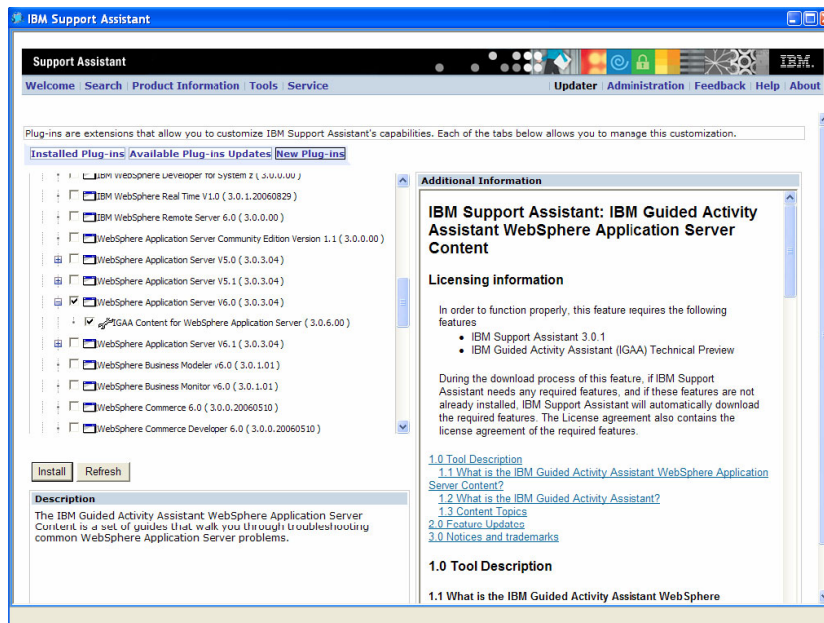


Figure 3-6. Installing the plug-ins

WA5711.0

Notes:

The IBM Support Assistant supports product plug-ins for most IBM products, across the brands. For example, there are product-plug-ins for DB2, WebSphere Portal Server, Rational Application Developer, CICS, MQ Server, and many others.

The WebSphere Application Server product plug-in for v6.1 includes data collection scripts that map to the WebSphere Level 2 *MustGathers* and address problem types such as:

1. Out of memory problems
2. Hang problems
3. High CPU problems
4. Security problems
5. Deployment manager and node agent discovery problems
6. Database pooling problems
7. Crash problems

- 8. Synchronization problems between the deployment manager and node agent
 - 9. Start and stop problems
 - 10. EJB problems
 - 11. SOA/Web services problems (multiple scripts for engine, tooling, security, and UDDI problems)
 - 12. JMS problems
- Plus many others.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Installing the tools

1. Select **Updater->New Plug-ins**
2. Expand **WebSphere** and select **WebSphere Application Server V6.1**
3. Select **IGAA Content for WebSphere Application Server**
4. Click **Install**

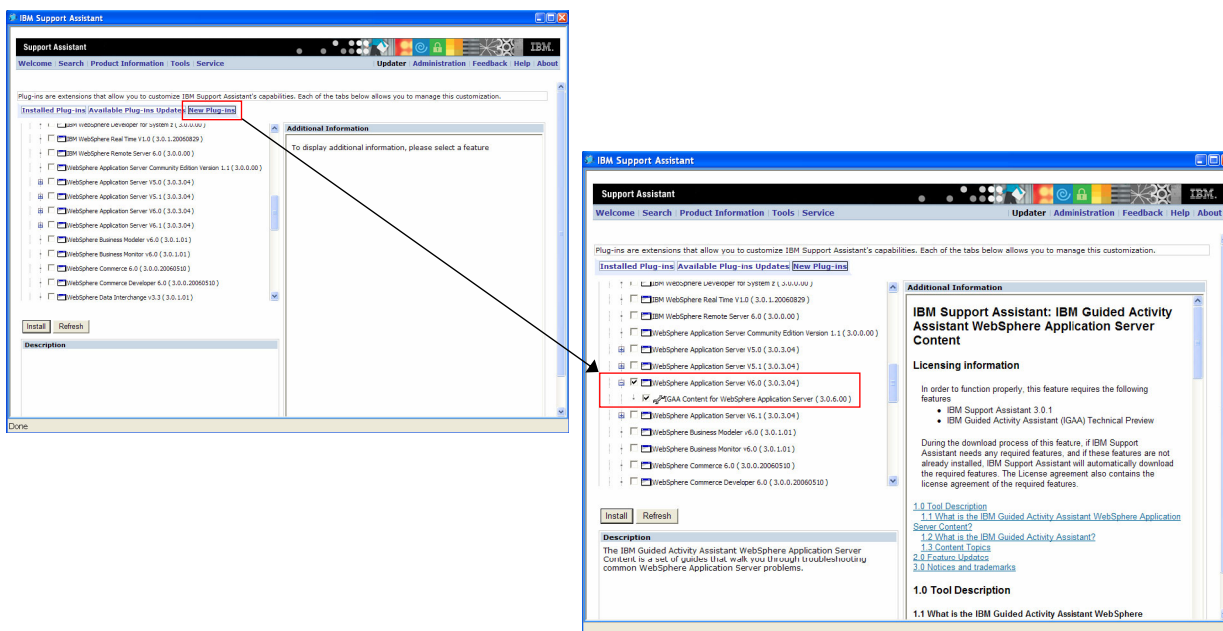


Figure 3-7. Installing the tools

WA5711.0

Notes:

Clicking the “New Products and Features” link takes you to a list of product plug-ins and tool plug-in organized by IBM software group brand. Selecting the product name displays additional information about the plug-in. The checkboxes will remain checked between update sessions; therefore, it might be necessary to uncheck a plug-in or tool that you do not want to install during a subsequent Updater session.

Instructor notes:

Purpose —

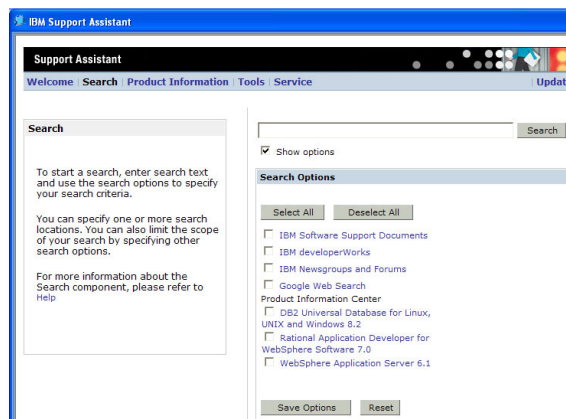
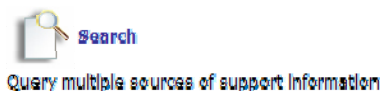
Details —

Additional information — Problems can arise when an error occurs during an update and the student attempts the update, again. The UI might be in a state where an undesired check box is checked, resulting in problems and confusion. Therefore, it might be necessary to explicitly uncheck these checkboxes and start again.

Transition statement —

Search component

- Multiple search engines can be queried simultaneously
 - Both IBM and external search engines are available



- The IBM Support Assistant *Help* link contains a section on using the Search component
 - The semantics of the search field is based on the various search engines
 - For example, the plus (+) character will create a phrase from a group of words

Figure 3-8. Search component

WA5711.0

Notes:

Using Search Options

Search options are provided to help you narrow the scope of your search. The search location you choose determines what search options are available. For example, if you select Google Web Search, search options do not apply. The IBM developerWorks and IBM Newsgroups and Forums search locations let you narrow your search to specific product areas.

The IBM Software Support Documents search location offers the most options. You can select specific products and versions, and you can select specific types of documents. The online IBM information centers are only available if they have been enabled within a product. Not all products will have an IBM information center to search.

Instructor notes:

Purpose —

Details —

Additional information — The search semantics are not fully documented in the Help system. Determining how to use advanced query features, such as word grouping, is dependant on the search engine that is integrated into ISA. ISA is a front end to several search engines – not a replacement for them. Therefore, if using a '+' in *Google* has a different meaning then using it in the WebSphere information center, then each of these search engines will interpret the query based on their semantics.

Transition statement —

Performing a search

- The search panel is highly configurable
 - Internal and external search engines, products, and product versions can be used as input criteria
- The various search engines will be contacted and queried using the parameters and search string supplied
 - The combination of parameters used for a given search engine will vary, depending on the search engine semantics

The screenshot shows the search interface with the following details:

- Search input: **SSL Exception**
- Search button: **Search**
- Show options:
- Search Options panel:
 - Select All / Deselect All buttons
 - Tree view:
 - IBM Software Support Documents
 - any document type
 - specific document type
 - IBM Downloads
 - IBM APARs
 - IBM Books and Articles
 - IBM Technotes
 - IBM developerWorks
 - any product family
 - specific product family
 - IBM Newsgroups and Forums
 - any product family
 - specific product family
 - Google Web Search
 - Product Information Center
 - DB2 Universal Database for Linux, UNIX and Windows 8.2
 - Rational Application Developer for WebSphere Software 7.0
 - WebSphere Application Server 6.1
 - Limit search by products:
 - DB2 Universal Database for Linux, UNIX and Windows
 - Rational Application Developer for WebSphere Software
 - Rational XDE Developer
 - WebSphere Application Server
 - any version
 - specific versions
 - 6.1
 - WebSphere Studio Application Developer
 - WebSphere Studio Site Developer
 - Select All / Deselect All buttons
 - Save Options / Reset buttons

Figure 3-9. Performing a search

WA5711.0

Notes:

Search Options

Search options are provided to help you narrow the scope of your search. The search location you choose determines what search options are available. For example, if you select Google Web Search, search options do not apply. The IBM developerWorks and IBM Newsgroups and Forums search locations let you narrow your search to specific product areas.

The IBM Software Support Documents search location offers the most options. You can select specific products and versions, and you can select specific types of documents.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Search results

- After the search is complete, click any of the links in the search results tree to view the list
- Click any item in the list to read the details

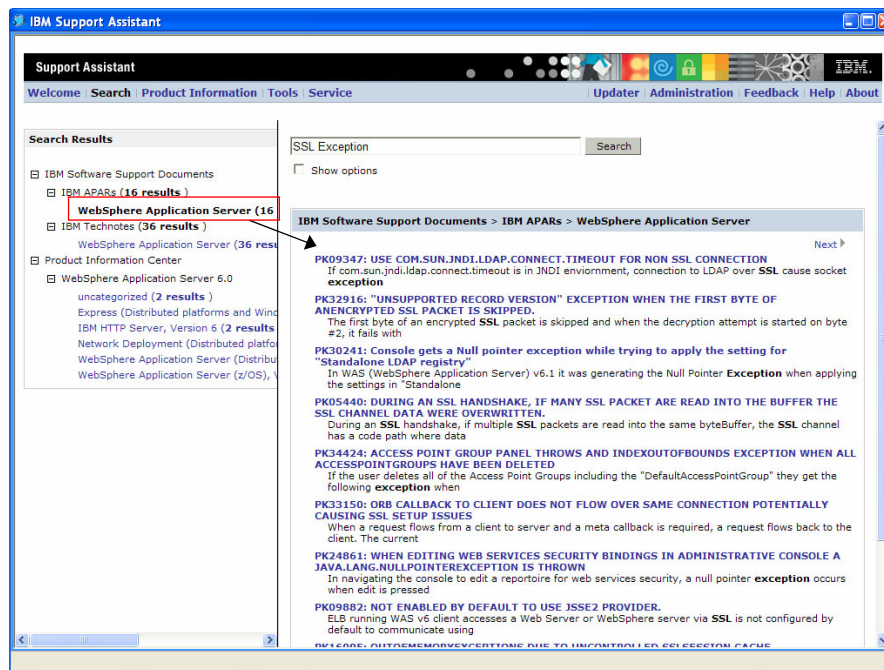


Figure 3-10. Search results

WA5711.0

Notes:

Clicking the links in the results list will bring up a Web browser in which you can read the document containing the search string.

The use of word separators, such as the + character will result in searches that match a complete phrase, rather than any of the words. For example, *Java core file* will search on *Java* and *core* and *file* while *Java+core+file* will search on the complete phrase for document matches.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Product Information component

- Select a product to produce marketing, support, skill enhancement, and problem solving resources integrated into a single view

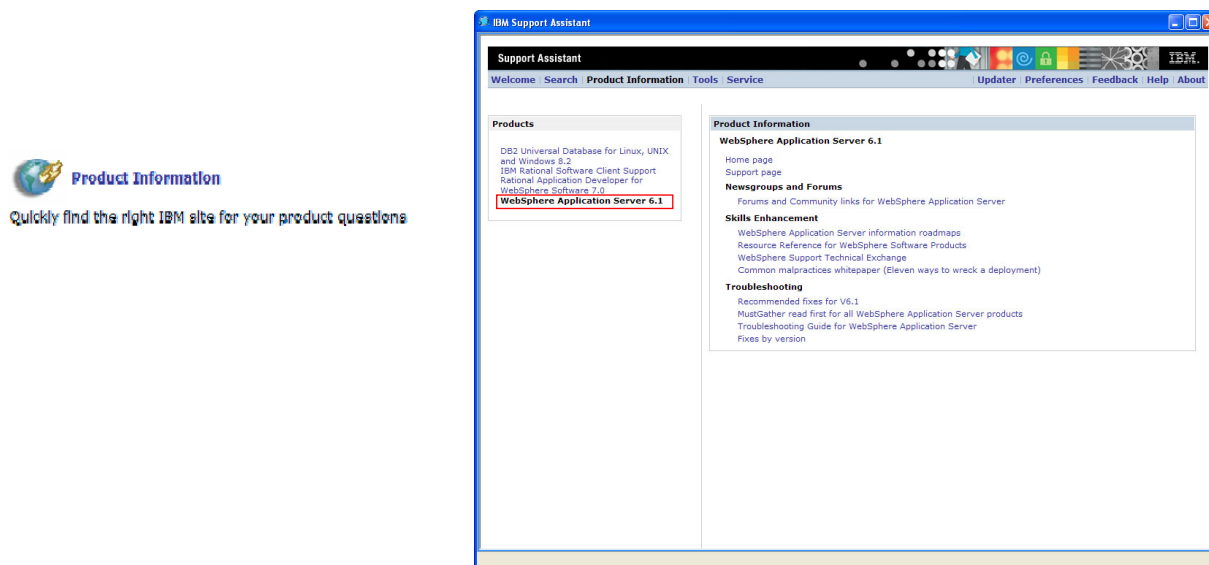


Figure 3-11. Product Information component

WA5711.0

Notes:

Product Information

The *Product Information* panel creates a navigational relationship between the selected product and the resources available for that product. The various URLs that relate to the product are grouped together to form an integration of the marketing, support, educational, and problem determination information.

Typical categories include:

- The product home and support page
- Newsgroups and Forums
 - These news groups and forums are typically monitored by IBM product experts and can be extremely useful when troubleshooting a problem
- Skills Enhancement

- The skills enhancements include educational resources that are available for download as well as classes that are being offered
- Troubleshooting
 - In addition to the fixes and troubleshooting guides, this section contains a link to the *MustGather* documents. The MustGathers contain step-by-step information for resolving a problem. The data collection scripts are based on these MustGathers.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tools component

- Tools component
 - Provides support tools to identify and prevent problems
- Use the Updater feature to download the latest tools
 - The tools are available through the Updater panel



Tools

Investigate product problems using specialized analysis tools

- Each product will have its own set of tools
 - Some tools are common to multiple products

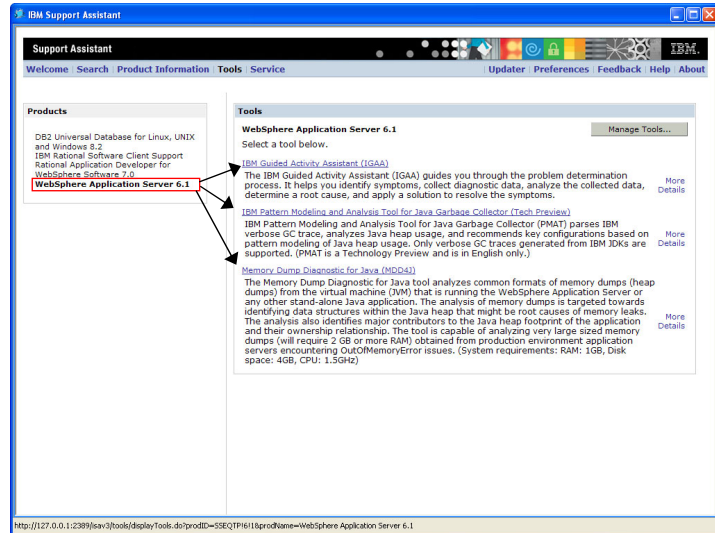


Figure 3-12. Tools component

WA5711.0

Notes:

The Tools component contains various problem determination tools that assist with processing the trace, verbose GC, and heap dumps that are produced from running ISA data collections. A tool can be installed, using the *Updater*, but without a product plug-in, that is related to the tool, the tool will not be visible. For example, the *IBM Guided Activity Assistant*, the *IBM Pattern Modeling and Analysis Tool for Java Garbage Collector*, and the *Memory Dump Diagnostic for Java* tools are associated with the WebSphere Application Server V6.1 product plug-in. From the *Tools* panel, clicking the WebSphere Application Server V6.1 link, in the left hand *Products* menu, will display the tools in the *Tools* frame of the workbench.

Instructor notes:

Purpose —

Details — Make the points that:

- The ISA workbench is initially void of product plug-ins and tools. In order to use the ISA they will need to install the features through the Updater
- The Updater should be used, regularly, to upgrade or add new functionality to the ISA workbench

Additional information —

Transition statement —

IBM Guided Activity Assistant (IGAA)

- This tool provides a step-by-step guide for:
 - Configuration tasks
 - Troubleshooting
- Helps gather:
 - Problem information
 - Diagnostics
- Helps analyze the gathered data to solve the problem

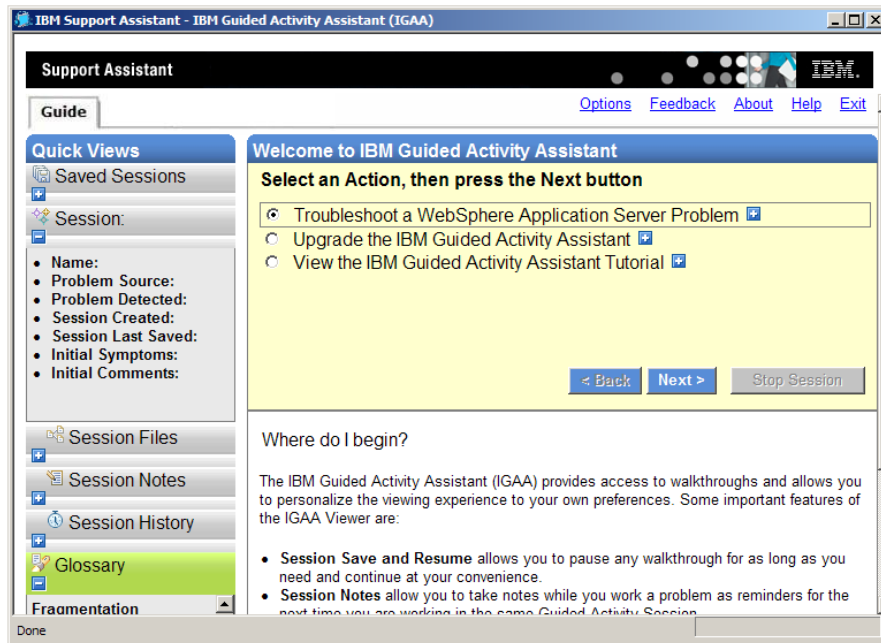


Figure 3-13. IBM Guided Activity Assistant (IGAA)

WA5711.0

Notes:

The IBM Guided Activity Assistant (IGAA) provides access to walkthroughs and allows you to personalize the viewing experience to your own preferences. Some important features of the IGAA Viewer are:

- **Session Save and Resume** - Pause a guided activity
- **Session Notes** - Information can be recorded during a problem determination session and used as a reminder in future guided activities based on the same problem
- **IGAA Upgrades** - An update feature that is specific to IGAA. This feature updates the specific pieces of the guided activities, rather than upgrading to a newer version of the plug-in

Select the **View the IBM Guided Activity Assistant Tutorial** radio button to become familiarized with the tool, prior to performing an initial guided activity.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Service component

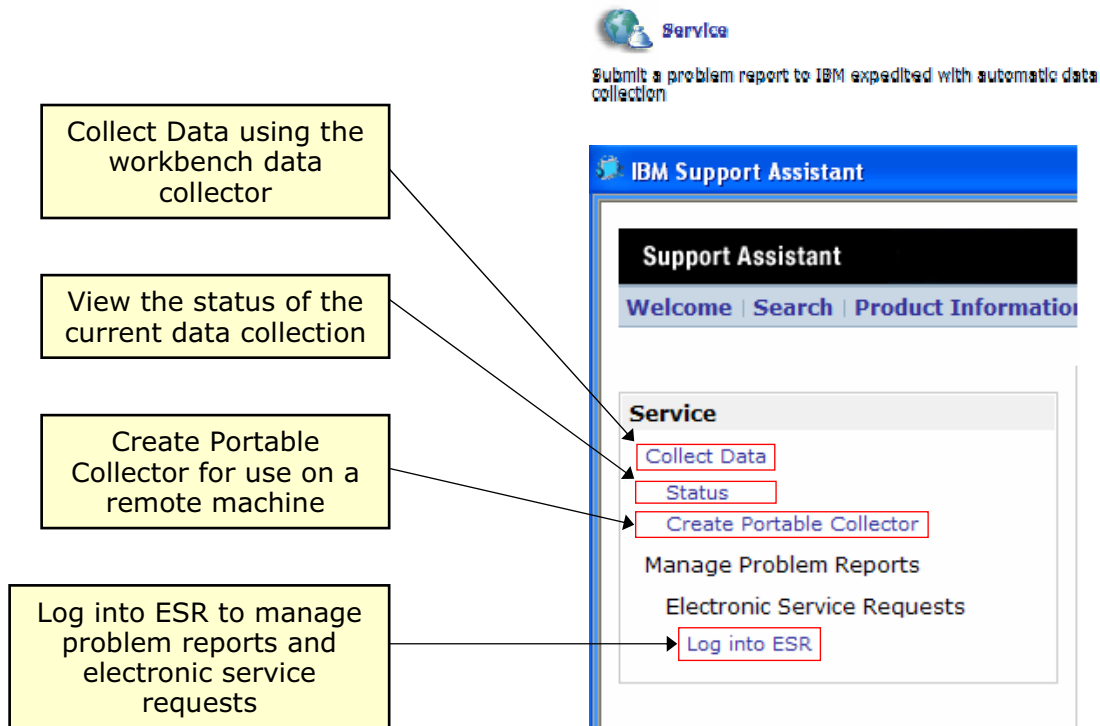


Figure 3-14. Service component

WA5711.0

Notes:

Data Collection

Collecting data is an important part of submitting a problem report because it enables IBM to resolve your problem report more quickly. The IBM Support Assistant supports data collection through the Eclipse-based workbench or as a command line driven portable collector.

Problem Report Management

In order to manage problem reports, you need to log in. Once you are logged in, you can;

- Open a new problem report
- View existing problem reports
- Attach a file to a new or existing problem report.

- To log in, you need to provide your IBM ID, password, IBM customer number, and country. The login process also verifies that you have an IBM service contract and that your ID is on the Authorized Callers list.
- After you log in, the Service component shows you any problem reports you have already submitted. You can see these problem reports at any time by clicking on List Problem Reports.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Data collection scripts

- WebSphere Application Server v6.1 contains a set of scripts that are based on the level 2 support *MustGathers*
 - The *MustGathers* are available from the WebSphere Application Server V6.1 support page
- From the service panel, the product plug-in is selected resulting in a drop-down list containing the problem categories
 - New plug-ins are being added on a regular basis
 - It is important to check for new or updated plug-ins using the IBM Support Assistant Updater

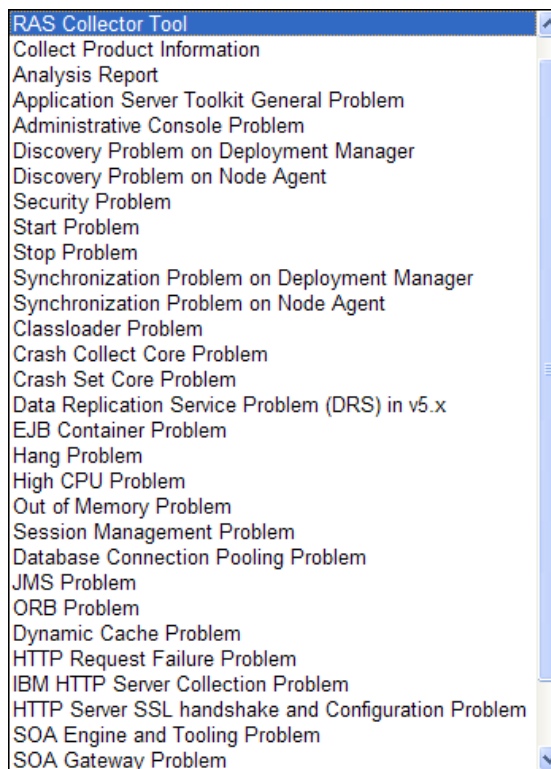


Figure 3-15. Data collection scripts

WA5711.0

Notes:

Submitting Problem Reports

Click *Submit Problem Report* if you would like to create a new problem report. To create a new report, you will need to answer a few basic questions:

- The product that is creating the problem
- The severity of the problem
- The nature of the problem
- Recent changes
- Problem resolution attempts
- Operating system information

By using the *System Collector*, that is integrated in ISA, or a product specific data collector the product and environment information needed for the problem report is automatically gathered and archived. Once the problem report and data gathering is complete, click *Submit* to send your problem report to IBM along with any associated files.

Existing problem reports can also be updated through ISA by clicking on *List Problem Reports* and then *Attach File* to add new information to the report.

Instructor notes:

Purpose —

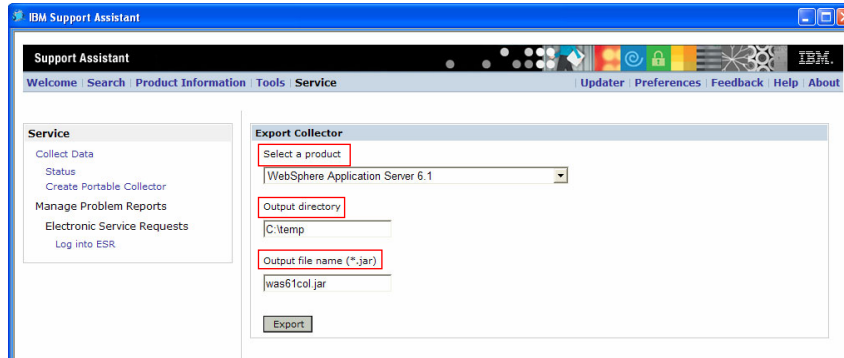
Details —

Additional information —

Transition statement —

Creating a portable collector

- Select **Create Portable Collector** from the Service menu on the left
 - Select the product from the **Select a product** drop-down menu
 - Enter the target output directory
 - This is the directory that the portable collector will be exported to, not where the collection will necessarily take place
 - Enter an output file name in the **Output file name** text field
 - It must include a .jar extension



- Once the portable collector has been exported, it is transferred to the enterprise host machine using FTP

Figure 3-16. Creating a portable collector

WA5711.0

Notes:

The portable collector is the typical method of data collection in an enterprise environment. Some of the scenarios used to deploy the portable collector include:

- Hosting it on a shared file system so it can be mounted when a data collection is required
- Deploying the portable collector to a sub-set of the nodes in a cell or cluster so it can be isolated to a few machines
- Performing in-house verification and re-packaging it for use by the IT department

The portable collector is a command line tool that has exactly the same level of functionality as the collector hosted in the ISA workbench. It is exported as a Java Archive (JAR) file and is transferred to a host using FTP or some other means. The portable collector creates an output file that can be transferred back to the ISA workbench and used as input to the problem determination tools or attached to a PMR.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Running the portable collector

- The portable collector is a command line tool that can run on multiple platforms, regardless of the platform used to generate the portable collector
 - Prior to running the tool the following actions are performed
 - Un-jar the contents of the portable collector
 - `jar xvf <portable collector name>.jar`
 - Set the JAVA_HOME for the environment
 - On UNIX, `export JAVA_HOME=<path to jre>`
 - On Windows, set `JAVA_HOME=<path to jre>`
 - On UNIX, change the permissions on the collector files
 - `chmod -R 755 *`
 - Enter the command `startcollector.[sh | bat]`

```
</tmp/collector> export JAVA_HOME=/usr/java14/jre
</tmp/collector> chmod -R 755 *
</tmp/collector> ./startcollector.sh
logger params set to
Start collector tool .....
AUTOPD_HOME is /tmp/collector
JAVA_HOME is /usr/java14/jre
Enter the Output Filename/Path (must be a local path, and have a .zip file extension):
(example: <pmrNumber>.<branchNumber>.<countryCode>.<desc>.zip):
```

Figure 3-17. Running the portable collector

WA5711.0

Notes:

The type of system and log data collected depends on the problem type you select.

The collected data is archived in a time stamped jar file

- Some applications provide a facility so the user can specify the location and name of the archive, using the Problem Management Report expression
 - For example: `/tmp/55555.333.333.general.0951_0768.jar`

Analysis reports may be generated in an HTML file

- For example: `autopd_analysis_report.html`

Some problem types make use of Automated PD

- Enable tracing of relevant components on a specific server
- Automatic stop/restart of server
- Option to reproduce problem and have trace output collected

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Output from Collect Data and Automated Problem Determination

- Examples of jar file and autopd analysis report from running the system collector

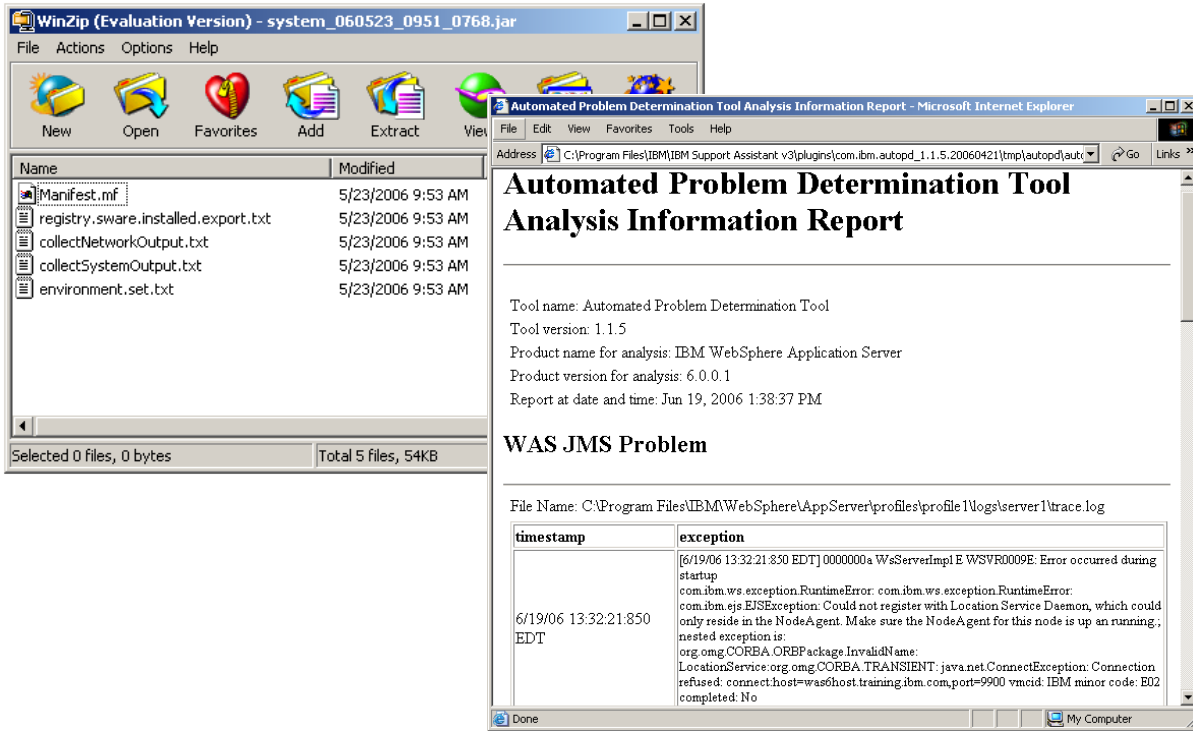


Figure 3-18. Output from Collect Data and Automated Problem Determination

WA5711.0

Notes:

The collection archive contains the configuration, log, trace, and analysis files. The type and amount of information collected will depend on the problem script that was chosen during the data collection.

Throughout this course there will be various modules where it is necessary to collect data from WebSphere Application Server V6.1 in order to determine the root cause of the problem and perform the resolution. It is important to keep in mind that any of these data collection activities can be performed using the IBM Support Assistant and one of its data collection scripts.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe the different support components such a Search, Product Information, Service, and Tools
- Recognize when to use the IBM Support Assistant
- Update IBM Support Assistant with new software plug-ins
- Install new support tools
- Use Collect Data and Automated Problem Determination (PD) features
- Use the IBM Support Assistant for problem determination

Figure 3-19. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 1: A guided tour of using IBM Support Assistant

Figure 3-20. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 4. Problem determination methodology

Estimated time

01:00

What this unit is about

This unit describes the problem determination methodology used in this course.

What you should be able to do

After completing this unit, you should be able to:

- Understand a problem
- Devise a plan of action
- Carry out the plan
- Examine the solution

How you will check your progress

Accountability:

- Checkpoint

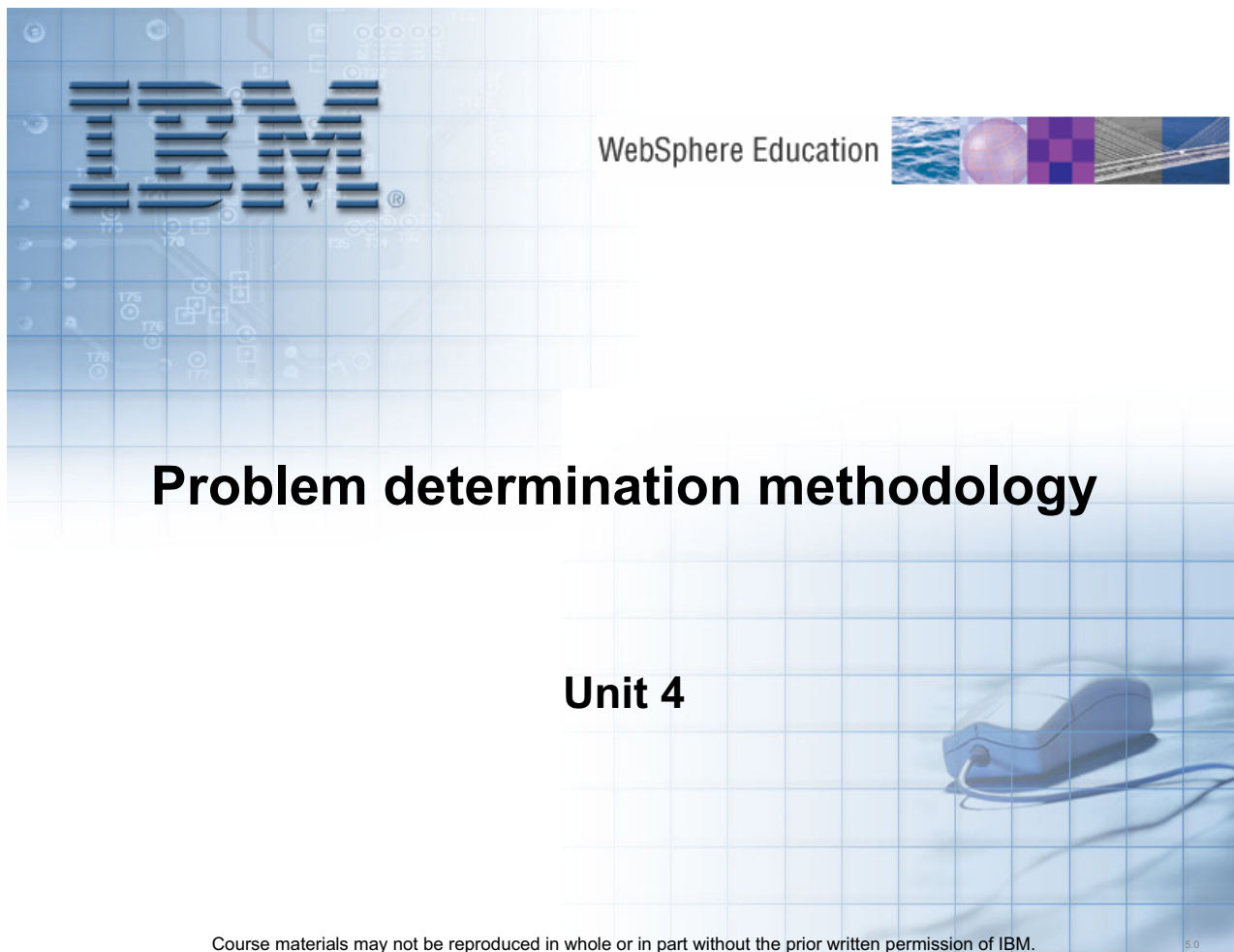


Figure 4-1. Problem determination methodology

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Understand a problem
- Devise a plan of action
- Carry out the plan
- Examine the solution

Figure 4-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

General observations about problem determination

- Problem determination is not an exact science
- Problem determination is not “rocket science”
- Problem determination is often a cooperative and iterative process
- The biggest obstacle to problem determination is poor communication
- Not every problem requires the most complex problem determination skills and techniques

Problem determination is not magic, but a matter of common sense, thoroughness, and clear communication.

Figure 4-3. General observations about problem determination

WA5711.0

Notes:

A major challenge of problem determination is dealing with unanticipated problems. It is much like detective work: find clues, make educated guesses, verify suspicions, and so on.

The most important skills are common sense, focus, thoroughness, rigorous thinking.

Customer self-service and automation for common problem determination tasks is available.

Instructor notes:

Purpose —

Details — Use some examples to illustrate the points on this slide. Describe how problem determination is an iterative process. Describe how issues can arise from poor communication.

Additional information —

Transition statement —

Types of problem symptoms

- Here are the common types of symptoms that you might see. Almost every symptom falls into one of these categories:
 1. Cannot install or migrate WebSphere Application Server or install an application into WebSphere Application Server.
 2. Experience difficulties in WebSphere Application Server system management or configuration.
 3. An application or WebSphere Application Server process (for example, an application server, node agent, or deployment manager) is unable to start.
 4. An application does not respond to incoming requests.
 5. An application produces unexpected results (possibly errors or exceptions).
 6. An application cannot connect to an external system or resource.
 7. An application performs slowly or its performance degrades over time.

Figure 4-4. Types of problem symptoms

WA5711.0

Notes:

Most problems are due to:

- Environment and configuration issues
- Misunderstandings or miscommunication
- Hard-to-diagnose application issues, and so on

Many problems are seen more than once at different customer sites.

Most problems are solved fairly quickly and easily.

A few problems take more time to resolve and pose a difficult challenge to the problem determination process.

Most problems are not related to product code defects. Non-defect oriented problems (NDOP) are typically greater than 90% of PMRs. Rediscoveries (both DOP and NDOP) are typically greater than 50% of PMRs. Some problems take more time to resolve because they are hard to diagnose, have never been seen before, are critical to production, involve

issues in the WebSphere Application Server internals, or may involve multiple interdependent issues, and so on.

Also, the approach to problem determination can be different, depending on if the symptoms are seen in a development versus production environment.

Instructor notes:

Purpose — The desired message to convey here is that problem determination is not magic, but rather a matter of common sense, thoroughness, and clear communications. The methodology outlined here is intended to help the student achieve the required communication skills, and so on.

Details —

Additional information —

Transition statement —

Key steps for problem determination

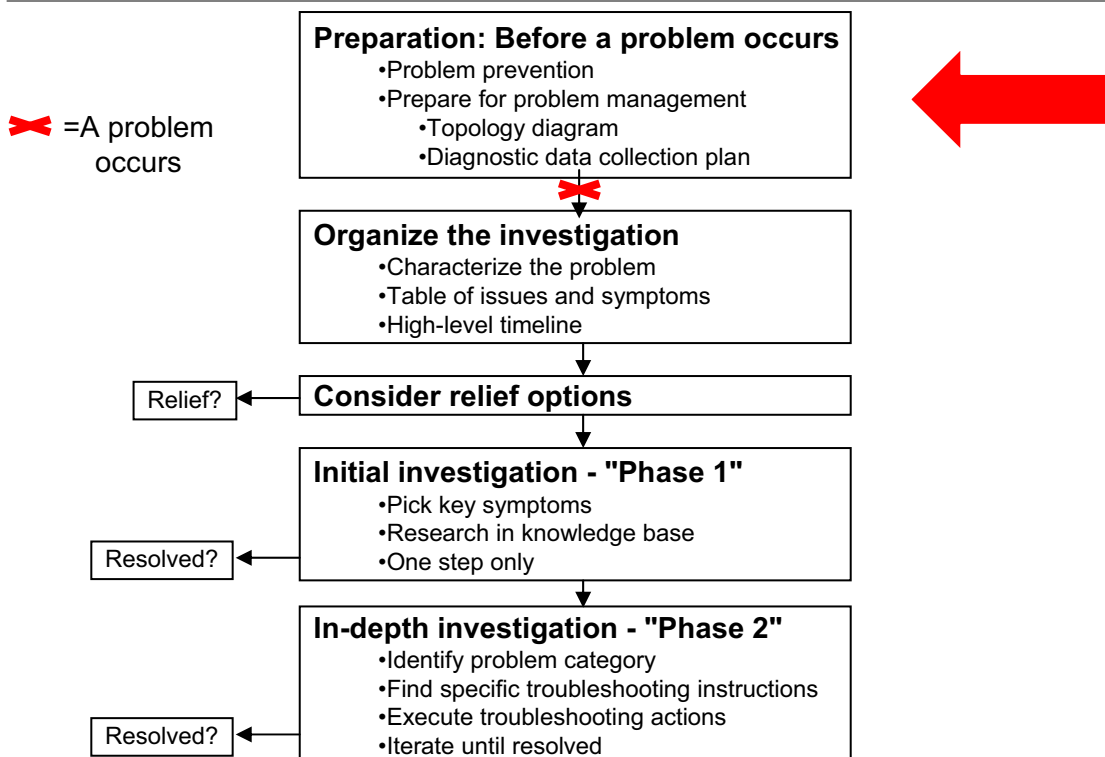


Figure 4-5. Key steps for problem determination

WA5711.0

Notes:

The red X represents the point at which a problem occurs. The job of a troubleshooter does not begin after a problem occurs. There is plenty of preparation work that should be done well before a problem occurs, if one is to run a good IT shop and be able to do good problem determination when it becomes needed.

One pitfall that many troubleshooters fall into is, they see a problem (for example, system is crashed), and they dive into doing all kinds of long and complex analysis, maybe for days and weeks. Meanwhile, the system is down and users are seeking short-term relief. Some relief options are discussed later in this presentation.

Relief is something that one should keep in mind at all times throughout the problem determination process, not just once as suggested in this chart. In any troubleshooting situation, you have three goals:

- Quickly provide a temporary solution so that users who are affected by the problem can get back to work, while you spend more time looking for the permanent solution
- Find and implement the right, permanent solution

- Try to make sure that a similar problem will not occur again in the future, or that if it does, you will be as prepared to deal with it as possible, after what you have learned from this problem

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Preparation: Before a problem occurs

- Good problem determination starts long before anything bad happens
- Implement problem prevention best practices
- Perform monitoring and problem detection
- Keep good system documentation
- Have a diagnostic data collection plan
- Have a relief or recovery plan
- Keep a maintenance plan: Scheduled and emergency
- Keep a change log

Figure 4-6. Preparation: Before a problem occurs

WA5711.0

Notes:

Problem prevention best practices:

- Providing a sufficient test environment
- Doing load or stress testing
- Capacity planning
- Keeping the system operating within the capacity plan
- Having a production traffic profile (network, and so on)
- Having a process for rolling-out changes into production
- Keeping a record of changes
- Doing application review and best practices
- Providing education
- Having a migration plan
- Having a current architecture plan

Monitoring: Often, problems go undetected for a long time, or only get detected indirectly through some secondary effect. You need tools and a plan to effectively detect problems or any potential anomaly when they emerge.

System documentation: When investigating a problem, you will need a lot of information about the system, and especially about how the system works when there is no problem. This information should be collected in advance. There are two key devices for formalizing this information:

- A topology or flow diagram
- A series of baselines for the normal behavior of the system

Diagnostic data collection: When a problem does occur, you need to gather information quickly and effectively that will allow you to diagnose the problem. To do this, you need a plan set up *in advance* for what diagnostics you will collect and how.

Relief or recover plan: When a problem occurs, one of the main priorities, independent of any investigation, should be to restore function to the end-users. The relief or recovery plan lays out, in advance, the steps that you will undertake to accomplish this, without knowing in advance exactly what problem will occur.

Maintenance: Applying regular maintenance is one of the key factors to reduce the probability and impact of problems. The maintenance plan establishes how you will do this on a regular basis. In addition to regular scheduled maintenance, you may also need to perform emergency changes or maintenance to the system, in response to a newly-diagnosed problem. The emergency maintenance plan outlines how to do this safely and effectively.

Change log: Your log should at least track all upgrades and software fixes applied in every software component in the system, including both infrastructure products and application code. It should also track every configuration change in any component. Ideally, it should also track any known changes in the pattern of usage of the system; for example, expected increases in load, a different mix of operations invoked by users, and so on.

**Note**

These are covered in the reference material at the end of this unit.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitoring and problem detection (1 of 3)

- Often, problems go undetected for a long time, or only get detected indirectly through some secondary effect. You need tools and a plan to effectively detect problems or any potential anomaly when they emerge.
- Monitoring is a trade-off: You want to detect important events, and yet not adversely impact the normal operation of the system.
- Monitoring is an entire technical area in itself, different from problem determination. This course will only cover a few pointers.

Figure 4-7. Monitoring and problem detection (1 of 3)

WA5711.0

Notes:

Latent problems are not uncommon, so monitoring is important to consider. It means doing more than just noticing when service to users is interrupted.

The course *WF881: IBM WebSphere V6 Performance Monitoring and Tuning for Administrators*, offered by the WebSphere Education team, can be supplemental and complimentary to this course:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitoring and problem detection (2 of 3)

- **Passive monitoring**
 - At all levels: Network, operating system, WebSphere, application, dependent systems (for example: database, directory, and so on).
 - Monitor the main system log files for errors and events.
 - For example, detecting application server restarts
 - Monitors and alerts on key system metrics.
 - OS: Memory usage, CPU, and so on
 - WebSphere: Default PMI, Performance Advisors
 - A few tools:
 - Event Alerter (subscribes to JMX notifications from WebSphere Application Server)
 - Tivoli tools
 - Many ad-hoc scripts and tools for each installation
- **Active monitoring**
 - Beyond passive monitoring; the best way to make sure that everything is all right is to periodically test the operation of the system.
 - Localized pinging (for example, one server, one database connection, and so on).
 - End-to-end pinging: Periodically send an entire “dummy” transaction through the system, verify that it completes.
 - For example, using TMTP, Web-based load programs

Figure 4-8. Monitoring and problem detection (2 of 3)

WA5711.0

Notes:

Examples of ongoing system “health” monitoring:

- Significant errors in the logs emitted by the various components.
- Metrics produced by each component should remain within acceptable norms (for example, operating system CPU and memory statistics, WebSphere Application Server performance metrics, transaction rate through the application, and so on).
- Spontaneous appearance of special artifacts that only get generated when a problem occurs, such as Java dumps, heap dumps, and so on.
- Periodically send a “ping” through various system components or the application, verifying that it continues to respond as expected.

Examples of explicit actions to generate additional diagnostics:

- Actively trigger various system dumps, if they have not been generated automatically (such as Java dump, heap dump, system dump, WebSphere Application Server Diagnostic Provider dumps, or other dumps that might be provided by various products and applications). For example, when a system is believed to be “hung,” it is common

practice to collect three consecutive Java dumps for each potentially affected JVM process.

- Take a snapshot of key operating system metrics, such as process states, sizes, CPU usage, and so on.
- Enable and collect information from the WebSphere Application Server Performance Monitoring Infrastructure instrumentation.
- Dynamically enable a specific trace, and collect that trace for a given interval while the system is in the current unhealthy state.
- Actively test or "ping" various aspects of the system to see how their behavior has changed compared to normal conditions, to try to isolate the source of the problem in a multi-component system. For example:
 - Send an HTTP request directly to the application server, bypassing the Web server, or test some operations directly against a back-end database, bypassing the application server.
 - Test the responses from different WebSphere Application Server cluster members.
 - If applicable, test different functions of the application, to see if they are affected differently.
 - Selectively restart individual components of the application or the system.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Monitoring and problem detection (3 of 3)

- Watch low-level operating system and network metrics
 - Overall CPU and memory usage for the entire machine
 - CPU and memory usage of individual processes that are part of the application (or that the application depends on)
 - Paging and disk I/O activity
 - Rate of network traffic between various components
 - Check for reduction or total loss of network connectivity between various components
- Be prepared to actively generate additional diagnostics when a problem occurs

Figure 4-9. Monitoring and problem detection (3 of 3)

WA5711.0

Notes:

Examples of explicit actions to generate additional diagnostics:

- Actively trigger various system dumps, if they have not been generated automatically (such as Java dump, heap dump, system dump, WebSphere Application Server Diagnostic Provider dumps, or other dumps that might be provided by various products and applications). For example, when a system is believed to be "hung," it is common practice to collect three consecutive Java dumps for each potentially affected JVM process.
- Take a snapshot of key operating system metrics, such as process states, sizes, CPU usage, and so on.
- Enable and collect information from the WebSphere Application Server Performance Monitoring Infrastructure instrumentation.
- Dynamically enable a specific trace, and collect that trace for a given interval while the system is in the current unhealthy state.

- Actively test or "ping" various aspects of the system to see how their behavior has changed compared to normal conditions, to try to isolate the source of the problem in a multi-component system. For example:
 - Send an HTTP request directly to the application server, bypassing the Web server, or test some operations directly against a back-end database, bypassing the application server.
 - Test the responses from different WebSphere Application Server cluster members.
 - If applicable, test different functions of the application, to see if they are affected differently.
 - Selectively restart individual components of the application or the system.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

System architecture or topology diagram

- Show all the components and all the main flows in the system
 - Identify the various troubleshooting points in the system
 - where to find information or clues about the cause of a problem
- Clearly communicate between the various parties involved in the troubleshooting task, both inside your organization and when trying to explain a complex environment to IBM Support
- Answer and verify a favorite question of all troubleshooters: What has changed recently?
- Useful for:
 - Communicating with all parties involved in the problem determination effort
 - Identifying discrepancies between the expected environment and the current reality
 - Identifying points where monitoring or health checking can be done
 - Identifying points where diagnostics data can be collected
- Be specific and detailed
 - Indicate software versions, machine names, and IP addresses if possible
- Identify the main execution flows through this diagram

Figure 4-10. System architecture or topology diagram

WA5711.0

Notes:

One best practice is to meet with all the parties involved in the deployment and operation of the system together to draw or re-draw the system topology diagram.

- Often, the mere fact of having multiple groups together to compare their understanding of the topology of the system, yields very important clues and dependencies that the various groups might not have been aware of previously
- Then, you can work together, from the big picture, to decide where you should start looking for anomalies in the system that might cause the current problem

Instructor notes:

Purpose —

Details — -

Additional information —

Transition statement — An example of a topology diagram is on the next slide.

Example: Topology diagram

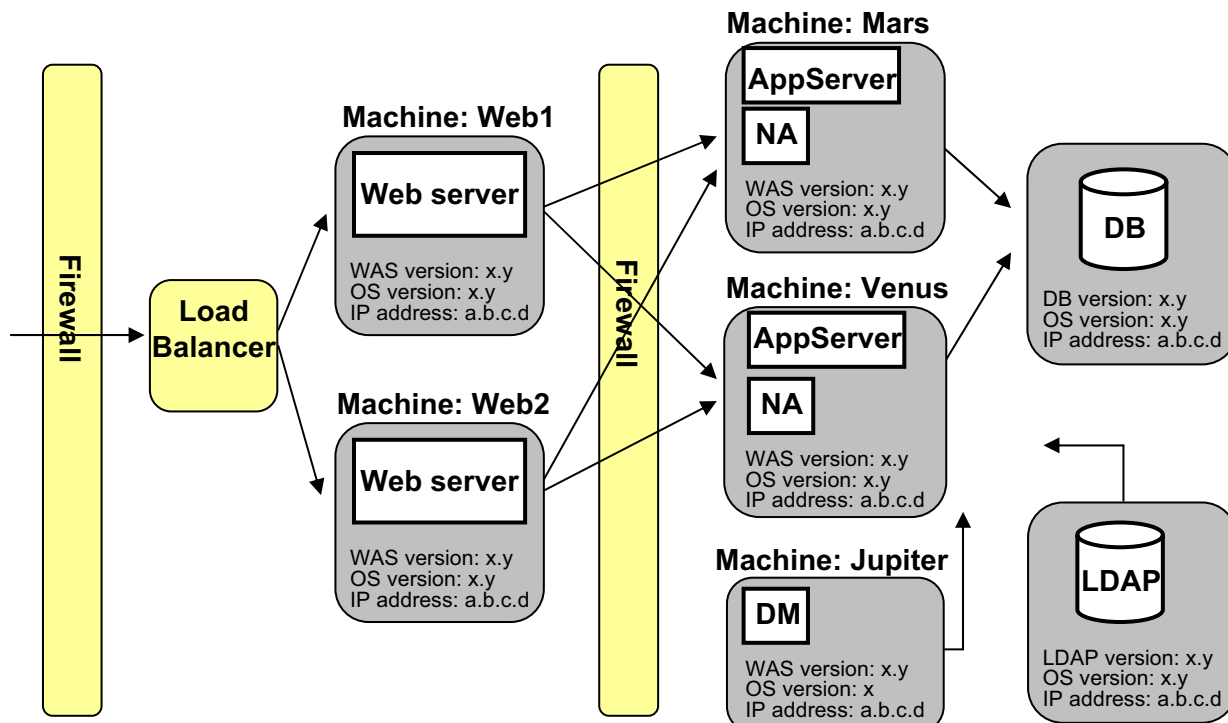


Figure 4-11. Example: Topology diagram

WA5711.0

Notes:

Some key elements to include in your topology diagram:

- Clearly show all the components in the system and their dependencies, not just the elements that are part of the main flow of processing
 - For example, administration services, security, and so on
- Be specific:
 - Machine names and IP addresses
 - Software versions of all software installed on each machine
 - Do not forget the network topology and relationships (a separate network topology chart may be useful)

This information will help form the basis for your diagnostic data collection plan.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Establish baselines

- Much of what you will do for problem determination is to observe various aspects of the behavior of the system and try to determine if these are normal or if they represent a symptom of a possible problem.
- Therefore, you must know before a problem occurs, what a “normal” system looks like.
- Some example criteria:
 - What are the typical values for common system metrics (CPU usage, memory size, PMI stats, and so on)?
 - These are often the same metrics that you should be monitoring for problem detection
 - What is a typical load, typical response time for various operations?
 - What should you normally see in the logs during normal operation of the system?
- Baseline information might have to be refreshed periodically

Figure 4-12. Establish baselines

WA5711.0

Notes:

In many actual systems, it is not unusual to see a variety of benign "errors" (both WebSphere Application Server and application-level) during normal operation. Learn to recognize them, or better yet, eliminate as many of these benign errors from the implementation of the system as possible.

Examples of baseline information:

- Copies of the various log files, trace files, and so on, over a representative period of time in the normal operation of the system, such as a full day.
- Copies of a few Java dumps, heap dumps, system dumps, or other types of artifacts that are normally generated "on demand." You can combine this activity with the earlier recommendation to test the generation of these artifacts on a healthy system before a problem occurs.
- Information about the normal transaction rates in the system, response times, and so on.

- Various operating system level statistics on a healthy system, such as CPU usage for all processes, memory usage, network traffic, and so on.
- Copies of any other artifacts, information or normal expected results from any of the special diagnostic collection actions, recommended earlier, for each anticipated type of problem.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Problem determination artifacts

- Make an inventory of all the important problem determination artifacts in your system
- The set of relevant artifacts will vary for each environment, set of products, and applications that you use
- Pay special attention to dumps and other artifacts that are only generated when a problem occurs
 - In most cases, there are a variety of configuration options that control how and when these artifacts are generated

Figure 4-13. Problem determination artifacts

WA5711.0

Notes:

Some of the most common artifacts are:

- All the standard logs files associated with WebSphere Application Server: activity.log, SystemOut.log, SystemErr.log, native_stdout.log, native_stderr.log, and so on.
- Incident files from the First Failure Data Capture (FFDC) facility in WebSphere Application Server.
- Log files from the Web server: access.log, error.log.
- Any log files from products built on top of WebSphere Application Server (such as WebSphere Portal, WebSphere Process Server, and so on).
- Any log files from other components that interact with the main application server, such as firewall logs, database server logs, and LDAP directory server logs.
- Any log files produced explicitly by an application.
- Log files and dumps produced by the Java virtual machine: javacore or Java dump, heap dump, and system dump (core file).

These problem determination artifacts will be covered later in the course.

For any file that can be generated automatically when a problem is detected: carefully consider the potential benefit and impact of having this file produced automatically, and set up the configuration accordingly. Do not leave this to chance. If the potential benefit is high and the impact is low, make sure that this feature is enabled.

For files that can be generated upon a specific action by an administrator: in most cases there is very little or no impact to the system until that particular action is taken. If at all possible, make the necessary preparations and configuration changes to ensure that the action will be available if and when needed, and test to make sure that it works as expected.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Diagnostic data collection plan (1 of 2)

- Decide in advance which diagnostic elements should be captured by default on the first occurrence of any problem
 - This is a trade-off: You want to capture as much as possible, without too much impact to the normal operation of the system
 - Some things to consider:
 - Default WebSphere logs
 - WebSphere First Failure Data Capture (FFDC) facility
 - HTTP access logs (what are default levels?)
 - Enable core dumps, Java dumps, and so on
 - Consider enabling verbose garbage collection
 - Consider a minimal level of monitoring of operating system resources
 - Consider a minimal level of monitoring of network health
 - Consider application logging

Figure 4-14. Diagnostic data collection plan (1 of 2)

WA5711.0

Notes:

Diagnostics to consider:

- JVM verboseGC log: Often very useful, and usually relatively low overhead on a well-tuned system.
- JVM Java dumps, heap dumps, and system dumps: Java dumps are typically somewhat cheap to produce, and can be enabled for automatic generation. Heap dumps and system dumps can involve significant overhead, so consider carefully before setting them up to be triggered automatically.
- Increased request logging at the HTTP server to show not just a single log entry for each request, but a separate log entry for the start and end of each request.
- A moderate level of monitoring performance counters provided by the WebSphere Application Server Performance Monitoring Infrastructure.
- Minimal WebSphere Application Server tracing to capture one or a few entries only for each transaction (Web requests or EJB requests).
- Application level tracing and logging, if any.

Verbose garbage collection has a very minimal impact on performance. The benefit usually outweighs the cost.

Each of these items will be explained later in the presentation, or discussed in detail later in the course.

Instructor notes:**Purpose —**

Details — Effective serviceability is always a trade-off. On one hand, to maximize the chances of being able to determine the cause of a problem upon its first occurrence, you want to gather the maximum amount of diagnostic data from the system at all times. But gathering very detailed diagnostics (for example, by having tracing enabled all the time) can cause a substantial performance overhead. Therefore, for performance reasons, you might be tempted to disable all diagnostics during normal operation of the system. You need to find the right balance between these two conflicting goals.

By default, most products and environments tend to err on the conservative side with a relatively small set of diagnostics enabled at all times. This is probably the right approach if no one will be available to review the set of enabled diagnostics before the system is put into production. As part of an actively designed troubleshooting plan, however, it is quite worthwhile to examine the specific constraints of your particular environment, the likelihood of specific problems, and the specific performance requirements, and then enable as many additional diagnostics as you can afford during normal steady-state operation.

Additional information —**Transition statement —**

Diagnostic data collection plan (2 of 2)

- Identify active diagnostic actions to take for the most likely problems. For example:
 - Trigger a Java core or core
 - Attempt pings and application checks
 - Collect specific PMI statistics
- Make arrangements so that all the pre-determined diagnostic elements get reliably captured
 - Clearly identify where they are
 - Established procedures to collect them and practice
 - Synchronize the clocks between all machines to facilitate analysis
 - Manage the logs and all other diagnostics artifacts to avoid surprises during normal operation
 - Make sure you have enough disk space available for diagnostic artifacts
- Do not forget the human element:
 - Who is in charge, when there is a production problem, of executing the diagnostic data collection plan and take recovery actions?
 - What groups need to be informed or coordinated with?

Figure 4-15. Diagnostic data collection plan (2 of 2)

WA5711.0

Notes:

These "active" diagnostic collection actions resemble those already mentioned when discussing regular monitoring (as in pinging). However, these are actions that you will perform only when you suspect a problem, as opposed to the regular monitoring. As such, you can afford to consider more invasive actions in this case.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Relief or recovery plan (1 of 2)

- When a problem occurs, one of the main priorities, independent of any investigation, should be to restore function to the end-users.
- The relief or recovery plan lays out, in advance, the steps that you will undertake to accomplish this, without knowing in advance exactly what problem will occur.
- It is impossible to predict all possible situations in advance, but on the other hand, much confusion, wasted time, and even greater disasters occur when relief actions are attempted “in the heat of the moment.” Preparation pays off.

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Relief or recovery plan (2 of 2)

- Try to predict the most common types of problems, based on knowledge of the system topology and flows.
 - For example, loss of one machine, loss of database connectivity, and so on.
- Identify different regions of the system that can be isolated or restarted independently (for example, one machine, one cluster, and so on).
- Have clear and documented processes for who will decide what to do and who will do it, and have criteria for making such decisions.
- Practice the most common relief actions in advance, to ensure that you know how to execute them, and that they have no unexpected side-effects.

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Maintenance plan

- Applying regular maintenance is one of the key factors to reduce the probability and impact of problems. The maintenance plan establishes how you will do this on a regular basis.
- In addition to regular scheduled maintenance, you may also need to perform emergency changes or maintenance to the system, in response to a newly-diagnosed problem. The emergency maintenance plan outlines how to do this safely and effectively.
- Maintenance is at all levels:
 - OS, and each of the products involved in the system.
- Keep track of current maintenance levels on the topology diagram.
 - Have processes in place for verifying the maintenance levels regularly and after each incident.

Figure 4-18. Maintenance plan

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Maintenance strategy: A word about fix packs

- IBM Support often recommends upgrading to the latest available fix pack during the course of an investigation.
 - <http://www.ibm.com/support/docview.wss?rs=180&uid=swg27004980>
- There is often a lot of concern about the risk of destabilizing the system by installing a fix pack.
- It is a difficult trade-off:
 - The risk can never be completely eliminated.
 - On the other hand:
 - There is also considerable risk in using too many individual fixes; no one can possibly test all the possible interactions between all individual fixes.
 - Because of the complexity of the system and the difficulty of reproducing problems and gathering diagnostics, it is not always practical to determine exactly which APAR resolved a particular situation.
- Overall, the best bet is to have a strong maintenance strategy.
 - Regular (small) updates.
 - Reasonable re-testing before each update.

Figure 4-19. Maintenance strategy: A word about fix packs

WA5711.0

Notes:

Web page for recommended fixes:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27004980>

Instructor notes:**Purpose —****Details —** So far, a number of documents have been mentioned:

- Topology diagram
- Monitor plan
- Diagnostics collection plan
- Relief plan
- Regular maintenance plan
- Emergency maintenance plan

You may want to review at this point.

Additional information — In some cases, installing the smallest fix possible is desirable. Installing multiple fixes at once can be risky. Some combinations of fixes are tested, but not all possible iterations. Sometimes, fixes will conflict with one another, especially if they are far apart or if too many are applied at once.**Transition statement —**

Key steps for problem determination

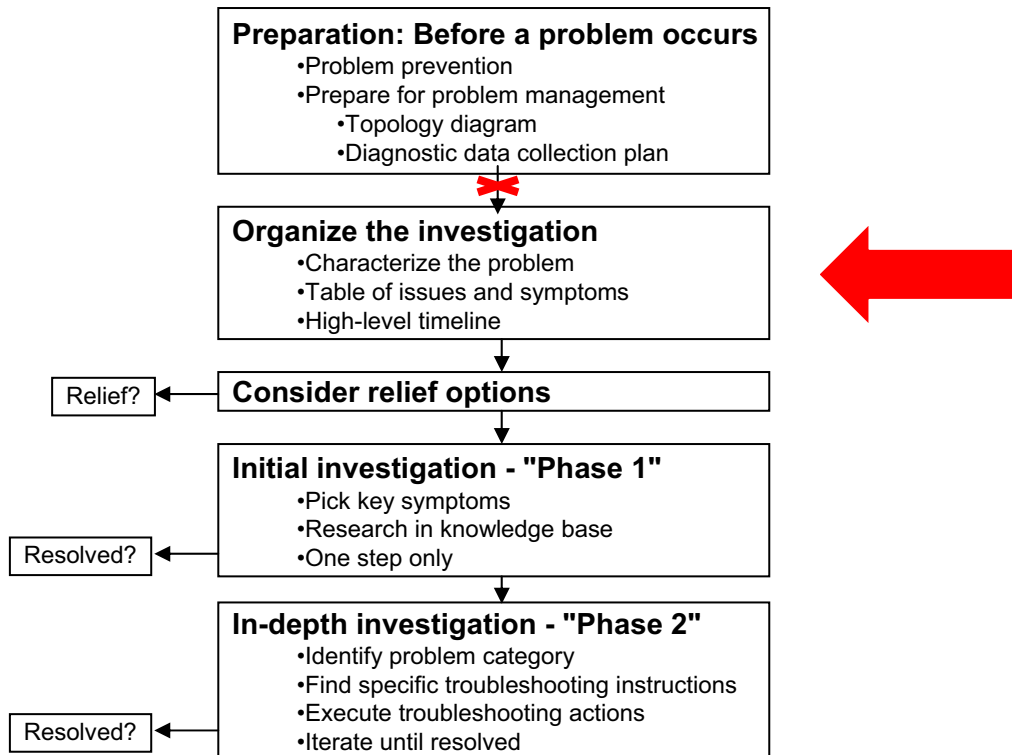


Figure 4-20. Key steps for problem determination

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Characterize the problem (1 of 2)

- Take the time to carefully understand the problem and its context. Listen and ask questions.
 - In many cases, this is all it takes to solve simple problems.
 - For complex problems, failure to do so often results in considerable delays
- Develop a clear and specific description of **what** happened, error messages, observed abnormal behavior, and so on.
 - How would you recognize the same problem if it happens again?
 - What exactly would you expect to be different once the problem is solved?
 - Beware of vague terms like *crash* or *fails*.
 - A *crash* is not the same as a *hang*, is not the same as an *exit*, and so on.
 - Be alert for possibly-unrelated symptoms, and the possibility that there may be several independent problems happening at once.
- **Where** exactly did the problem occur, what machine, what server, and so on

Figure 4-21. Characterize the problem (1 of 2)

WA5711.0

Notes:

There is a very good summary taken from G. Polya, "How to Solve It", 2nd ed., Princeton University Press, 1957, ISBN 0-691-08097-6, here:

<http://www.math.utah.edu/~alfeld/math/polya.html>

It explains a basic methodology for problem solving.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Characterize the problem (2 of 2)

- **When** exactly did the problem occur?
 - Did it happen only once, or does it happen repeatedly?
 - If the problem happens repeatedly, characterize the circumstances
 - Apparently random times? What frequency?
 - Every time you do X, or every time some other external event happens?
- Consider **why** this problem occurred here and now, and has not occurred before.
 - Is this the first time that you attempted something new (for example, first product installation)?
 - If not, has anything at all changed in the environment (configuration changes in the failing system or any other system that is also present in the environment)?
 - Does this problem occur in all similar environments (multiple production systems) or not?

Figure 4-22. Characterize the problem (2 of 2)

WA5711.0

Notes:

In many cases, you may be surprised how often careful consideration of these questions will get you very close to solving the problem, without even needing much further investigation.

And, in the rest of the cases, having given careful consideration to these questions may avoid much of the confusion, miscommunication and false starts that often plague complex problem investigations.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Key steps for problem determination

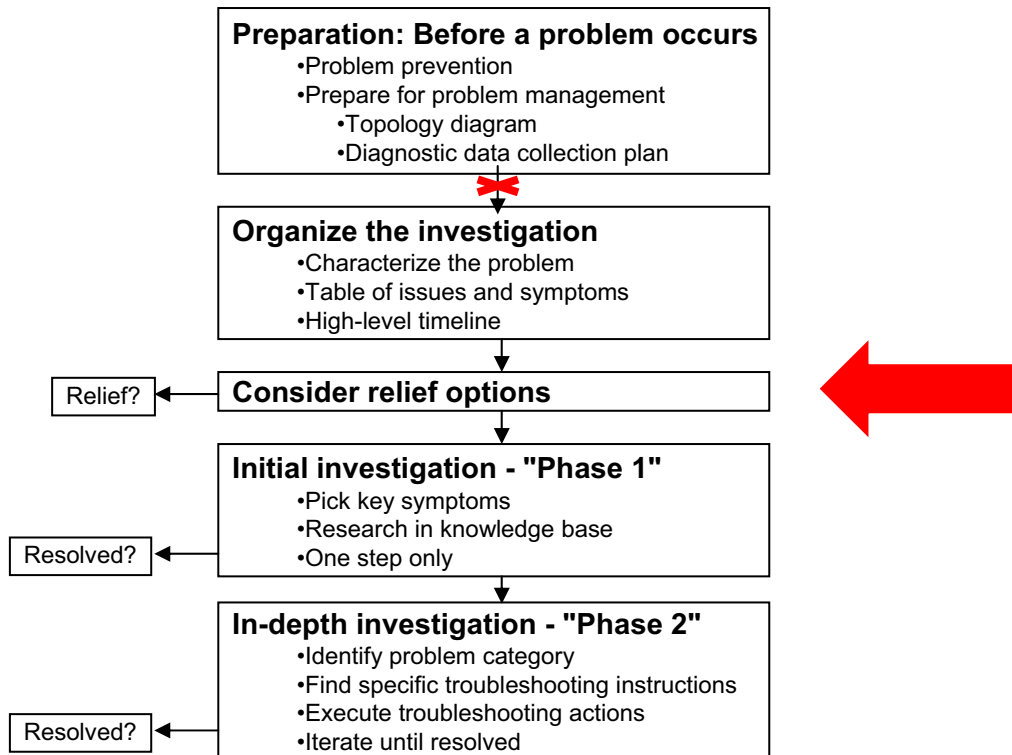


Figure 4-23. Key steps for problem determination

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Relief options

- If the problem occurs in production or other critical-availability system, you may need to provide relief long before you fully resolve the problem.
- Consider relief timeline:
 - How long do you have to investigate, before you need to provide some relief?
 - Can you keep the system in its failed state, in case there is some additional information that you want to collect later?
- Identify the relief actions:
 - Often, restart one or more components:
 - But beware of chain reaction effects of stopping and starting some components in a live system.
 - Sometimes, change the usage characteristics of the application:
 - Reduce the load.
 - Avoid some “dangerous” operations.
- Re-evaluate relief options at every step through the investigation.

Figure 4-24. Relief options

WA5711.0

Notes:

“Dangerous” operations refer to any function in the application that is deemed particularly likely to trigger a failure, maybe because it is more complex than others, or because it depends on more system components.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Key steps for problem determination

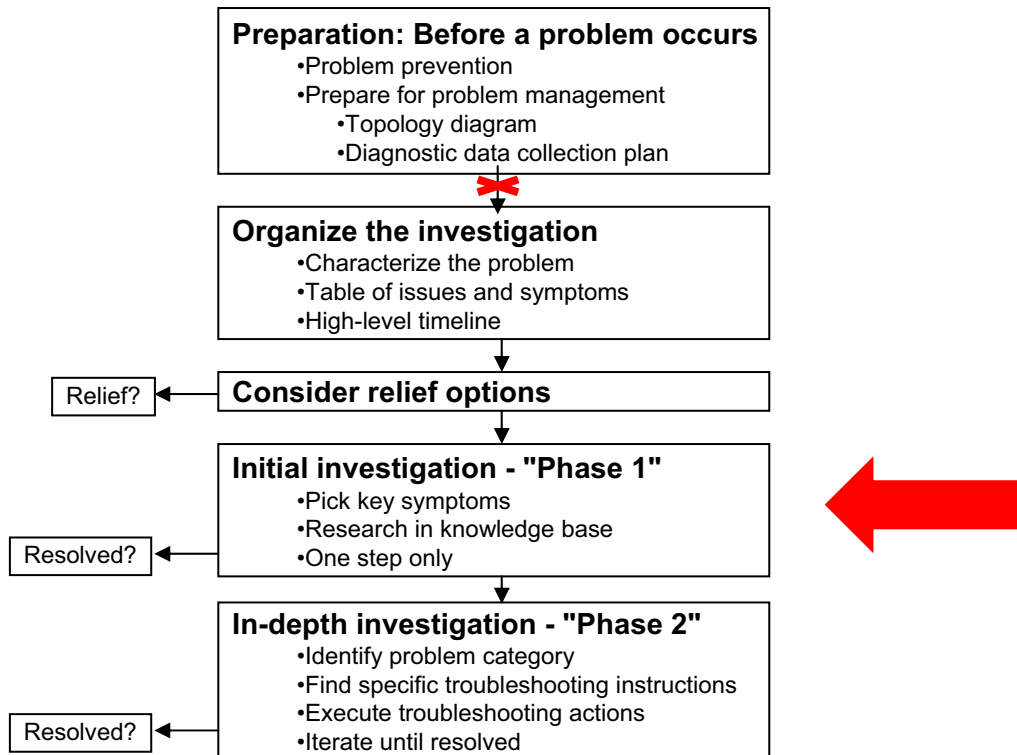


Figure 4-25. Key steps for problem determination

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

"Solve a problem" flow - big picture

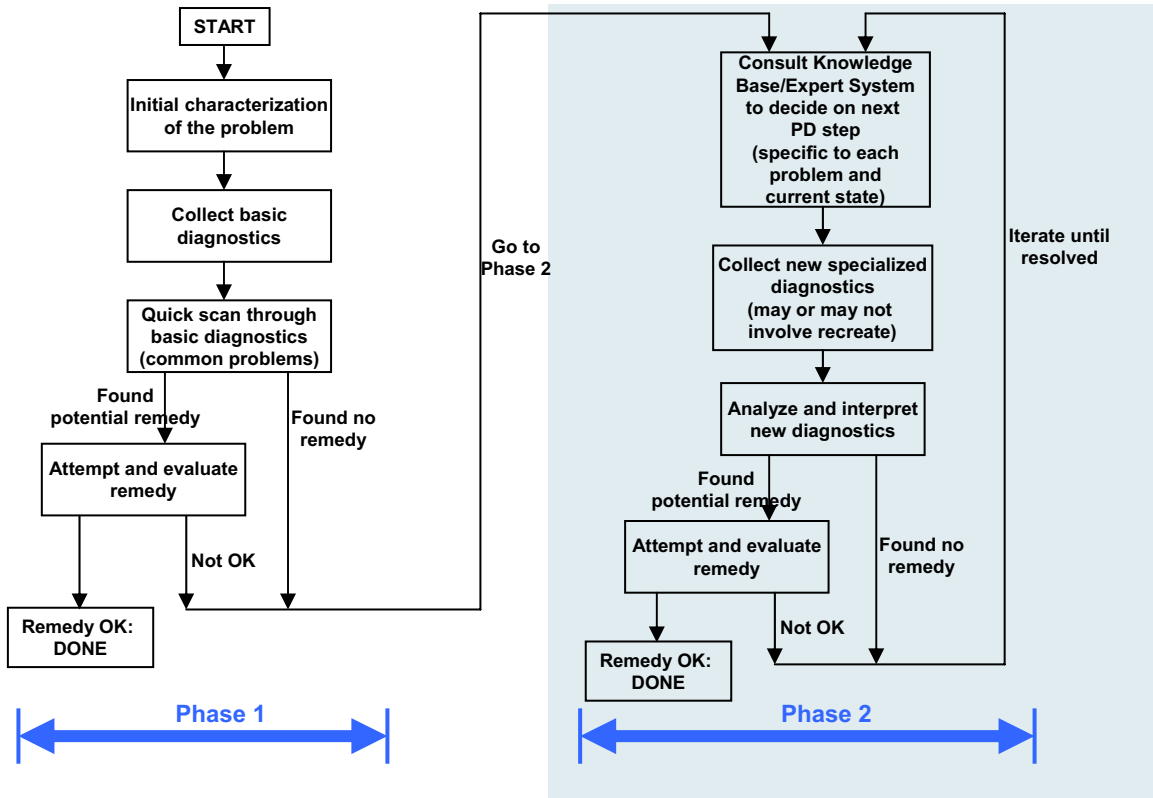


Figure 4-26. "Solve a problem" flow - big picture

WA5711.0

Notes:

This is not an “official” process, but for the purpose of this course, problem determination is broken down into two phases. Phase 1 represents the steps that are easy, obvious, or generic to every problem; Phase 2 gets into more depth and detail for a specific problem. This is where the process of resolution differs significantly for each problem.

Instructor notes:

Purpose —

Details — At the time that this course was developed this was not a widely used concept, but the idea is catching on.

Additional information —

Transition statement —

Phase 1: Initial investigation (1 of 2)

- The goal at this stage is to look for already known problems, and to get a starting point for further, deeper investigation if necessary.
- Make an inventory of all pertinent anomalies and potential symptoms:
 - Errors, warnings, exceptions, out-of-range statistics, any other unusual behavior. Scan (grep) through available logs.
 - Ideally, you should check everything, but this research might be guided by understanding the flows in the topology diagram.
 - Integrate into the table of issues and symptoms.
 - Assess and prioritize symptoms:
 - Not a perfect science; hard to tell which symptom is a cause, and which symptom is a consequence of the original problem.
 - Use the topology diagram and baselines prepared earlier.

Figure 4-27. Phase 1: Initial investigation (1 of 2)

WA5711.0

Notes:

Make sure that the problem is not a well-known issue. Do the research first.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Phase 1: Initial investigation (2 of 2)

- Build a low-level, detailed timeline of events for one incident, to clarify what may have caused what
 - Include pertinent but normal events, in addition to anomalies and symptoms
 - Keep track of multiple components in parallel and attempt to correlate
- Research the top symptoms in the WebSphere Knowledge Base
 - Search for matching Technotes
 - Search the Information Center
 - This will be covered in the next unit

Notes:

Building a low-level timeline will be covered later in the course. Also, searching the WebSphere Knowledge Base will also be discussed later in the course.

If you are lucky, at this stage, you find an article in the knowledge base that addresses the problem or symptoms you are seeing, and then you can apply the recommended solution.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Key steps for problem determination

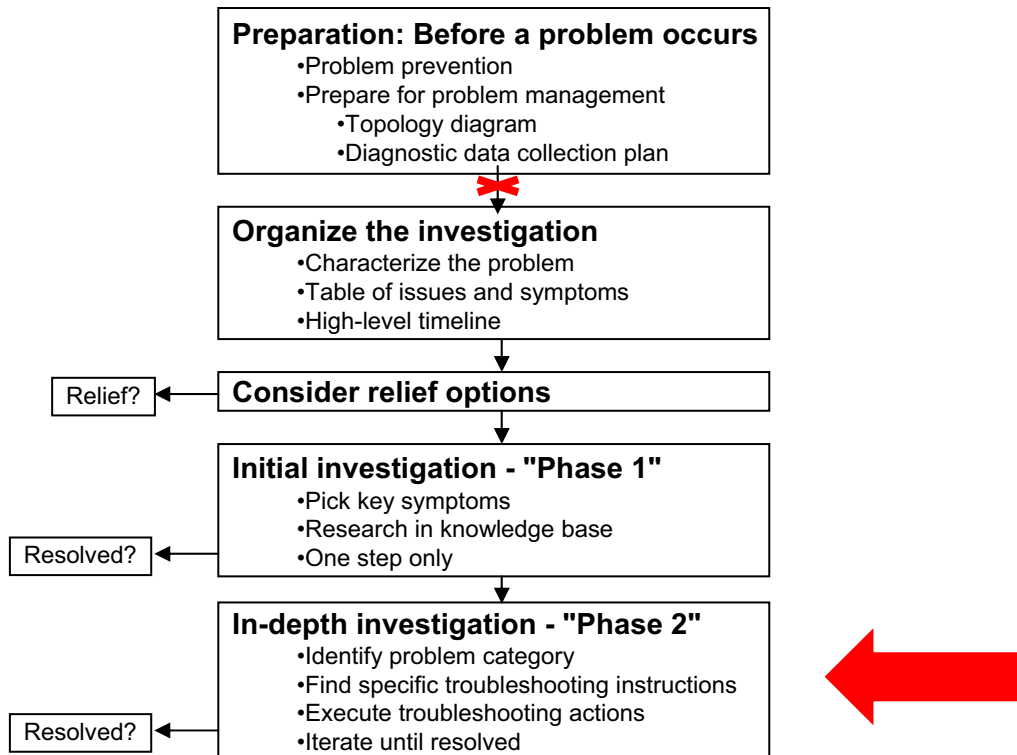


Figure 4-29. Key steps for problem determination

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

"Solve a problem" flow - big picture

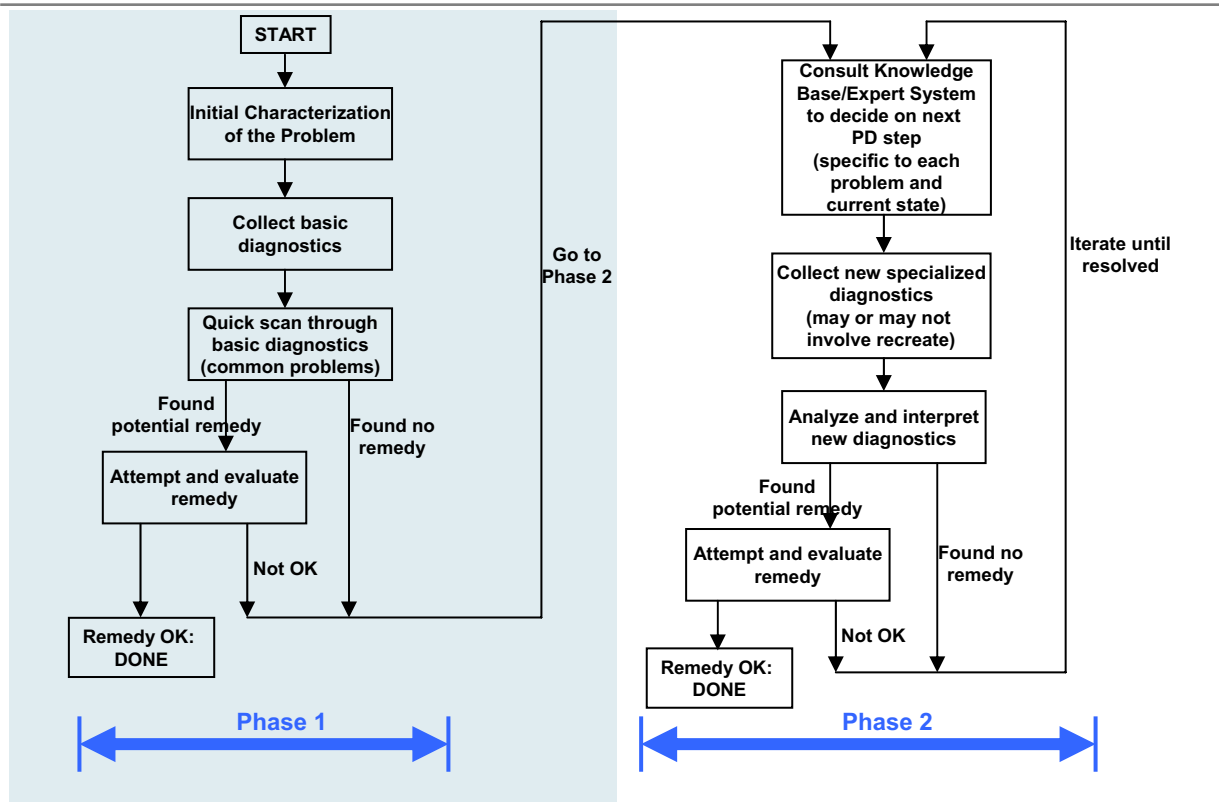


Figure 4-30. "Solve a problem" flow - big picture

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Phase 2: In-depth investigation (1 of 2)

- The initial research did not find a solution, so you must undertake a more in-depth investigation, with a specific set of methods and tools for each specific problem.

- There are several sources of information to start with:
 - Troubleshooting section in the InfoCenter
 - Problem determination IBM Redbook:
<http://www.redbooks.ibm.com/abstracts/sg246798.html?Open>
 - MustGather documents on the WebSphere Application Server Support Web site:
 - Define initial set of diagnostics to focus on (either for IBM Support or for internal investigation).
 - From there, search for other Technotes with troubleshooting instructions, tools, and so on.
 - The Troubleshooting Guide on the support Web site also provides a good starting point for finding relevant documents.

Figure 4-31. Phase 2: In-depth investigation (1 of 2)

WA5711.0

Notes:

In this phase, you are looking for specific instructions on how to solve a specific problem. Techniques for gathering information are covered in the next section.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Phase 2: In-depth investigation (2 of 2)

- The WebSphere Knowledge Base provides the starting point for the investigation process. What happens next depends on what you find.

- Two main investigation techniques:
 - Analysis approach
 - Use available diagnostic data in increasing detail until solution is found
 - Isolation approach
 - Isolate and simplify the problem to the smallest, simplest possible sub-unit of the original system, to reduce the amount of data to analyze

- In practice, these two approaches are often pursued in parallel, and feed into one another

- Consider if you might attempt to reproduce the problem in a test environment

Figure 4-32. Phase 2: In-depth investigation (2 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Example: Low-level timeline

	HTTDP Count	Mars Thread 1	Mars Thread 2	Mars Thread 3
12:00:00:***		RESTART -----		
09:30:00:***	10			
09:55:00:***	10			
09:59:10:564		> getConnection		
09:59:10:943		< getConnection		
09:59:15:134			> getConnection	
09:59:15:201				> getConnection
09:59:15:456		> getConnection		
10:00:00:***	125			
10:02:15:859			CONM6026W	
10:05:00:***	192			
10:10:00:***	193			
10:10:01:034			> getConnection	
10:10:01:750		GC allocation failure (2Meg) -----		
10:10:01:900		CRASH -----		

Figure 4-33. Example: Low-level timeline

WA5711.0

Notes:

This is a useful technique for doing “phase 1” investigation, but does not apply in every problem case.

If you are looking at a log or trace file for longer than five minutes, you may want to use a chart like this to organize your analysis. It may help you to keep track of certain events and when they occurred.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

For more information

- **IBM Redbooks:** www.redbooks.ibm.com
- Redbook SG24-6798-00: WebSphere Application Server V6 Problem Determination for Distributed Platforms
 - <http://www.redbooks.ibm.com/abstracts/sg246798.html>
 - Will be updated for V6.1
- Redpaper: WebSphere Application Server V6.1: Workload Management Problem Determination
 - <http://www.redbooks.ibm.com/abstracts/redp4308.html>
- IBM Redpaper draft: WebSphere Application Server V6.1: JMS Problem Determination
 - <http://www.redbooks.ibm.com/redpieces/abstracts/REDP4330.html>
- Articles on developerWorks:
 - Build a framework for problem determination Triage - <http://www.ibm.com/developerworks/autonomic/library/ac-pdtriage1/>
 - 12 ways you can prepare for effective production troubleshooting - http://www.ibm.com/developerworks/websphere/techjournal/0708_supauth/0708_supauth.html

Figure 4-34. For more information

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. Describe some preparation or prevention techniques that should be implemented before a problem occurs.
2. What are the key steps for problem determination?
3. What are relief options?

Figure 4-35. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1.
 - a.
 - b.
 - c.
 - d.
 - e.
 - f.
2.
 - a.
 - b.
 - c.

- d.
 - e.
- 3.
- a.
 - b.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

1. Describe some preparation or prevention techniques that should be implemented before a problem occurs.
 - **Implement problem prevention best practices**
 - **Perform monitoring and problem detection**
 - **Keep good system documentation**
 - **Have a diagnostic data collection plan**
 - **Have a relief or recovery plan**
 - **Keep a maintenance plan: Scheduled and emergency**

2. What are the key steps for problem determination?
 - a. **Preparation before a problem occurs**
 - b. **Organizing the investigation**
 - c. **Consideration of relief options**
 - d. **Initial investigation (phase 1)**
 - e. **In-depth investigation (phase 2)**

3. What are relief options?
 - **Actions you can take to address a problem while you are conducting a problem investigation**
 - **For example, restart one or more components, reduce load, and so on**

Figure 4-36. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Understand a problem
- Devise a plan of action
- Carry out the plan
- Examine the solution

Figure 4-37. Unit summary

WA5711.0

Notes:

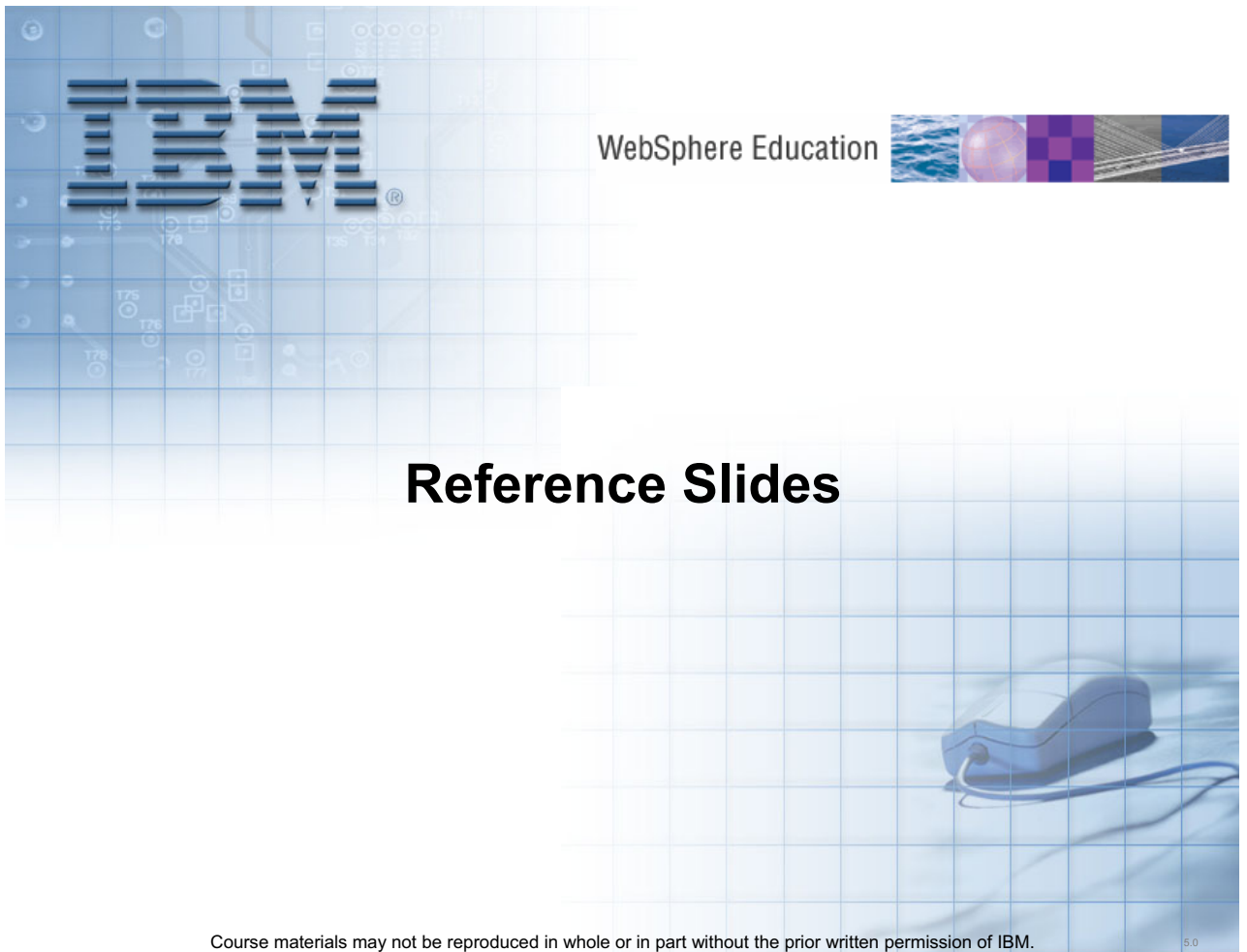
Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Figure 4-38. Reference Slides

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Review: Maintenance plan

- Applying regular maintenance is one of the key factors to reduce the probability and impact of problems. The maintenance plan establishes how you will do this on a regular basis.
- In addition to regular scheduled maintenance, you may also need to perform emergency changes or maintenance to the system, in response to a newly-diagnosed problem. The emergency maintenance plan outlines how to do this safely and effectively.
- Maintenance is at all levels:
 - OS, and each of the products involved in the system
- Keep track of current maintenance levels on the topology diagram
 - Have processes in place for verifying the maintenance levels regularly and after each incident

Figure 4-39. Review: Maintenance plan

WA5711.0

Notes:

This topic was discussed earlier in the course.

Viewing the fix level of the product

Use the `versionInfo` command and the `historyInfo` command in the `bin` directory of the installation root directory to display the exact fix and version level of the product. However, do not use either command while installing or uninstalling a maintenance package.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Review: Maintenance strategy

- IBM Support often recommends upgrading to the latest available fix pack during the course of an investigation.
 - <http://www.ibm.com/support/docview.wss?rs=180&uid=swg27004980>
- There is often a lot of concern about the risk of destabilizing the system by installing a fix pack.
- It is a difficult trade-off:
 - The risk can never be completely eliminated.
 - On the other hand:
 - There is also considerable risk in using too many individual fixes; no one can possibly test all the possible interactions between all individual fixes.
 - Because of the complexity of the system and the difficulty of reproducing problems and gathering diagnostics, it is not always practical to determine exactly which APAR resolved a particular situation.
- Overall, the best bet is to have a strong maintenance strategy.
 - Regular (small) updates.
 - Reasonable re-testing before each update.

Figure 4-40. Review: Maintenance strategy

WA5711.0

Notes:

Visit Recommended Updates to view a comprehensive list of recommended updates for WebSphere Application Server releases, along with a list of previously delivered updates.

Recommended updates Web page:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27004980>



Information

Receive a custom “My Support” e-mail each week with important news about the IBM products you select. My Support e-mail can now include technotes, release notes, education, and more. If you already receive weekly e-mails, update your profile today with the new options.

Get to My Support from any support page

- Open any Support page, including <http://www.ibm.com/software/support/>, and select **My support** in the box to the right
- or
- From the left column, select **Support** -> **My support** from the WebSphere software platform page at <http://www.ibm.com/websphere>

Instructor notes:

Purpose —

Details — -

Additional information —

Transition statement —

Making backups

- Each time changes are made to the configuration, use the **backupConfig** command
 - Creates a zip file that can be moved and stored offline
 - Use **restoreConfig** command to restore with a backup



Figure 4-41. Making backups

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Preventive measures during project life cycle

Stages	Best Practice	
Planning	Capacity or transaction plan Education Current architecture plan	*Good communication
Development	Application design and implementation	
Validation	Production traffic profile Load or stress testing Test environment; testing process	
Evolution	Migration plan Record of changes	

Figure 4-42. Preventive measures during project life cycle

WA5711.0

Notes:

Problem prevention best practices include the following:

- Providing a sufficient test environment
- Doing load or stress testing
- Capacity planning
- Keeping the system operating within the capacity plan
- Having a production traffic profile (network, and so on)
- Having a process for rolling-out changes into production
- Keeping a record of changes
- Doing application review and best practices
- Providing education
- Having a migration plan
- Having a current architecture plan

*Good communication among team members is essential at all stages of the life cycle.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Capacity or scalability plan

- Determine the amount of data that will flow into, through, and out of the application
- Determine the type and volume of data that will grow over time
- Determine the need for production capacity and response time
- Consider the length of each transaction
- Consider the number of concurrent users that will be in production
- Periodically update the plan as market changes

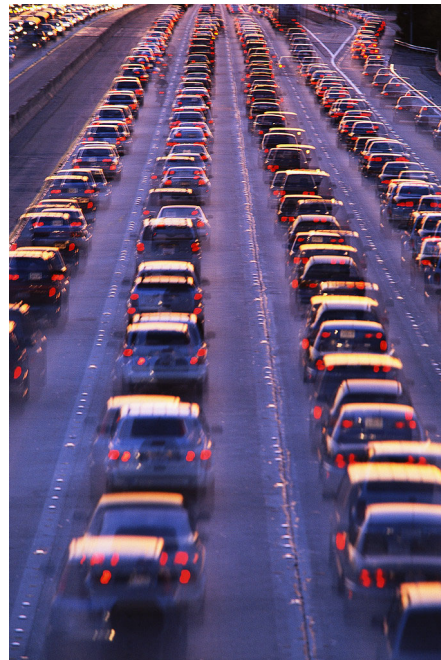


Figure 4-43. Capacity or scalability plan

WA5711.0

Notes:

While it is true that a capacity plan is not absolutely necessary, it is a plan that you need if the e-business ever needs to grow. The plan is not as simple as buying a bigger piece of hardware. The simple plan will work for a while, but it probably is not the most cost-effective way to increase the capacity of an application.

Use the following checklist to develop a capacity plan:

- What does the capacity plan for the back-end database say it can handle for an additional load?
- What can the network routers handle for increased network traffic?
- What can the Lightweight Directory Access Protocol (LDAP) server handle for user authentication?
- If the application is spread out across a Wide Area Network (WAN), are remote calls kept to a minimum?
- Is part of the hardware also being used for disaster recovery or HA?

- What is the maximum amount of hardware utilization before you need to provide additional hardware?
- Are object types being properly used for increased capacity?
 - Entity enterprise beans with a caching policy for the bulk of client database reads
 - Stateful enterprise beans have been passivated to a fast storage device

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Education

- Allocate time for team members to attend classes or study education materials on the following topics:
 - Problem determination
 - Performance tuning
 - Available product features
- Take time to learn about changes that occur with each new release of the product
- Allow time to review applications for the following changes:
 - Changes in business needs
 - Sections that are affected by J2EE capability changes

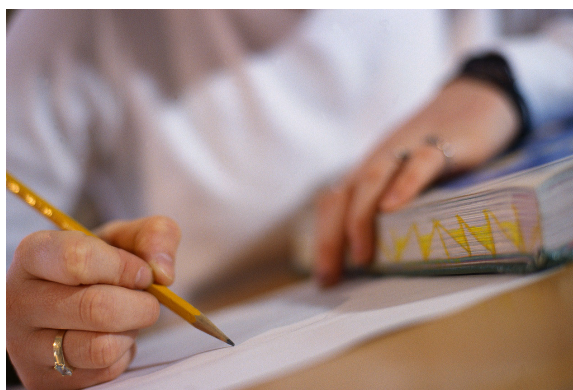


Figure 4-44. Education

WA5711.0

Notes:

The following education opportunities are all available free from IBM:

- IBM Education Assistant
<http://www.ibm.com/software/info/education/assistant/>
- Technical Exchange
http://www.ibm.com/software/websphere/support/supp_tech.html

Technotes and techdocs

- For WebSphere Application Server for Distributed platforms, see the following Web site:
<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27006233>
- For WebSphere Application Server for z/OS platform, see the following Web site:
<http://www.ibm.com/support/docview.wss?rs=404&uid=swg27006898>
- IBM Redbooks
<http://www.redbooks.ibm.com/redbooks.nsf/redpapers/>

- IBM Support Assistant
<http://www-306.ibm.com/software/support/isa/>
- Join the global WebSphere user group community:
<http://websphere.org>

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Current architecture plan

- A diagram of the application flow between the various software products
 - Shows where these products reside in the system topology
 - Current and up-to-date
- Answers the following questions:
 - Who changed what, when, and why?
 - What is the data flow?
 - What is the environment including both hardware and software?
 - What are the internal structures?

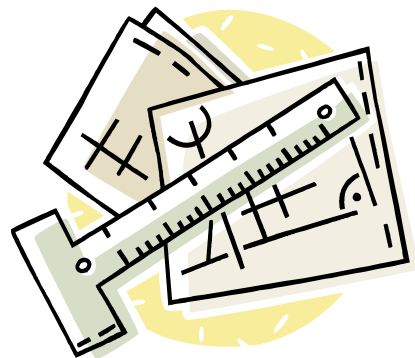


Figure 4-45. Current architecture plan

WA5711.0

Notes:

Additionally, you can use one of the architecture patterns that are recommended for the level of WebSphere Application Server that you are using. These architecture plans are defined in IBM Redbooks. The pattern recommendations might vary by WebSphere Application Server version level and do not absolve you of maintaining documentation for your own installation, but do provide a solid starting point. For more information on IBM Redbooks, see <http://www.redbooks.ibm.com/>.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Application design and implementation

- The development stage is the creation stage, which builds a foundation for the production stage. This stage is more than just the original development of the application, it also includes all of the enhancements or modifications to the application.
- Some common application errors to be aware of are:
 - Allocation of large objects causes the heap size to grow too big
 - Redundant computation calls causes increased CPU usage
 - Infinite loops cause increased CPU usage

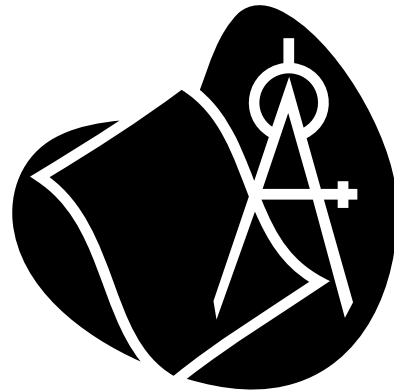


Figure 4-46. Application design and implementation

WA5711.0

Notes:

Ask yourself the following questions on a regular basis about the application life cycle:

- Does the application do what it is intended to do?
- Is the application designed using a modeling language?
- Is the application code-reviewed by someone other than the person who wrote it?
- Is the application serviceable?

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Production traffic profile

- A diagram of network, routers, switches, and hubs
- Includes data on the capacity of the networks or network segments
- You need to have the following information:
 - The location of the routers and firewalls
 - Where the servers are located on each network segment
 - How much traffic each server is expected to handle
- Ensure that there is sufficient bandwidth to handle the traffic

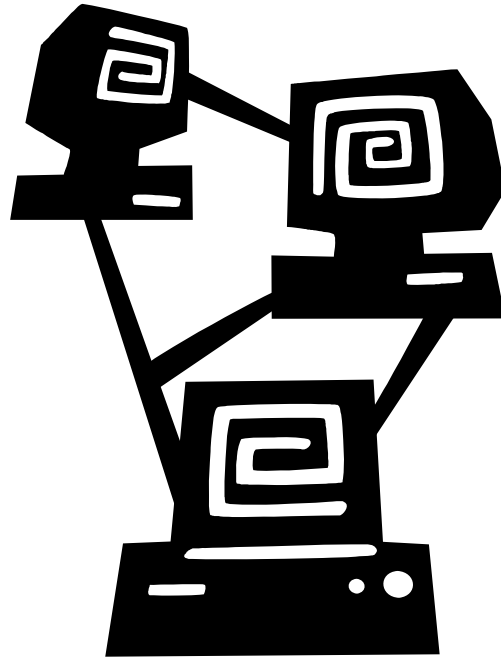


Figure 4-47. Production traffic profile

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Load or stress testing

- Do load or stress testing.
- Ideally, the load and stress tests involve every part of the application and simulate the real-world workload and user patterns.
 - Do load or stress testing based on the production load
 - Simulate the transaction pattern accurately to the production transaction pattern
 - Size the back-end database appropriately
- For existing applications, you must have a good understanding of their peak loads, duration, and their time of occurrence.
- When you do not know how a new application might be used, you must rely on the design document and common sense when you prepare the test script.



Figure 4-48. Load or stress testing

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Use a test environment

- Implement and test all changes in the test environment before moving into production
- Use the following checklist for the test environment:
 - Understand that a test system is a must, not a luxury; do not use a production system for test purposes
 - Physically separate a test system from a product system and ideally have two different groups of operators manage these systems
 - Clearly label the test and production systems with a different set of security identities
 - Install identical software on the test system as the production system; it is recommended that you use similar hardware for the test and production systems
 - Upgrade WebSphere Application Server and the applications themselves on the test system first before upgrading the production system
- Different types of test environments:
 - Maintain an exact duplicate of the production environment
 - Maintain a scaled-down environment with load generators that duplicate the expected load
 - Maintain a model or simulation of the environment

Figure 4-49. Use a test environment

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Migration plan

- Allocate adequate time for migration
- Investigate how new J2EE specifications affect the system
- Ensure that developers are aware of co-requisite software requirements
- Note the following:
 - Differences in default settings between versions
 - Differences in the vendor extensions
 - Changes to the operating system-specific configurations



Figure 4-50. Migration plan

WA5711.0

Notes:

Useful links for migration:

- WebSphere for z/OS Version 5.1 - Migrating Configuration from V5.0 to V5.1
http://www.ibm.com/support/docview.wss?rs=404&context=SS7K4U&dc=DA480&uid=tss1wp1_00441&loc=en_US&cs=utf-8&lang=en

Some work is required on your part prior to using the migration utilities for WebSphere Application Server. This site provides a concise, step-by-step explanation of the process that is used to migrate your configuration from version 5.0 to version 5.1. The "WebSphere 502 to 51 Migration Worksheet" document provides a set of concise data-capture worksheets. The worksheets are helpful tools to use when you are capturing the critical data that is required by the migration process.

- Migrating from WebSphere for z/OS V5.x to V6 - An example migration
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100559>

This white paper provides an illustration of an actual step-by-step migration so that you can see how it is accomplished and what some of the issues might be when you migrate. This

white paper is not the definitive source for migration information; rather, it is an example of one actual migration.

- IBM Education Assistant: WebSphere Application Server for z/OS V6.0.1 installation and migration

http://www.ibm.com/support/docview.wss?rs=404&context=SS7K4U&dc=DA480&uid=ss1wp1_00441&loc=en_US&cs=utf-8&lang=en

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Keep a record of changes

- Use a centralized change record system and maintain it accurately
- Create a strict process to inform every one of the changes
- Be aware of the fact that taking a little time to follow this process can prevent huge problems and embarrassment later



Figure 4-51. Keep a record of changes

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Facilitate good communication

- You must recognize the following communication barriers:
 - Installing products without considering product updates
 - Understanding the environment
 - Communicating with management
 - Considering cultural issues
 - Timing deployment
 - Coordinating the impending deployment with the affected products and services owners
 - Having a back out plan
- Use the following checklist to prevent communication breakdown:
 - Identify who needs to communicate on what subject and to what extent
 - Create a checklist to make sure the current information is synchronized
 - Have a periodic meeting to make sure that everyone understands the process
 - When someone is out, the backup person must know exactly what is occurring



Figure 4-52. Facilitate good communication

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Applying APARs

- If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support will create an Authorized Program Analysis Report (APAR).
 - The APAR describes the problem in detail
- Whenever possible, IBM Software Support will provide a workaround for you to implement until the APAR is resolved and a fix is delivered.
- IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 5. Problem determination tools and techniques

Estimated time

00:45

What this unit is about

This unit describes some of the common problem determination tools and techniques used in this course.

What you should be able to do

After completing this unit, you should be able to:

- Gather information for problem determination
- Identify and describe the main problem determination artifacts: logs, traces, dumps, PMI data, and so on
- Locate and describe the main WebSphere Application Server logs
- Enable basic tracing of the server and HTTP plug-in
- Check product versions and patch levels
- Apply maintenance (APARs)
- Locate and interpret WebSphere FFDC logs
- Use WebSphere PMI
- Search for information in the various knowledge bases for WebSphere Application Server Problem Determination information
- Enable and interpret WebSphere Application Server trace information (distinct from logs)
- Find and install the various tools that are available to assist in Problem Determination tasks
- Build a detailed low-level timeline of events for deep analysis

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

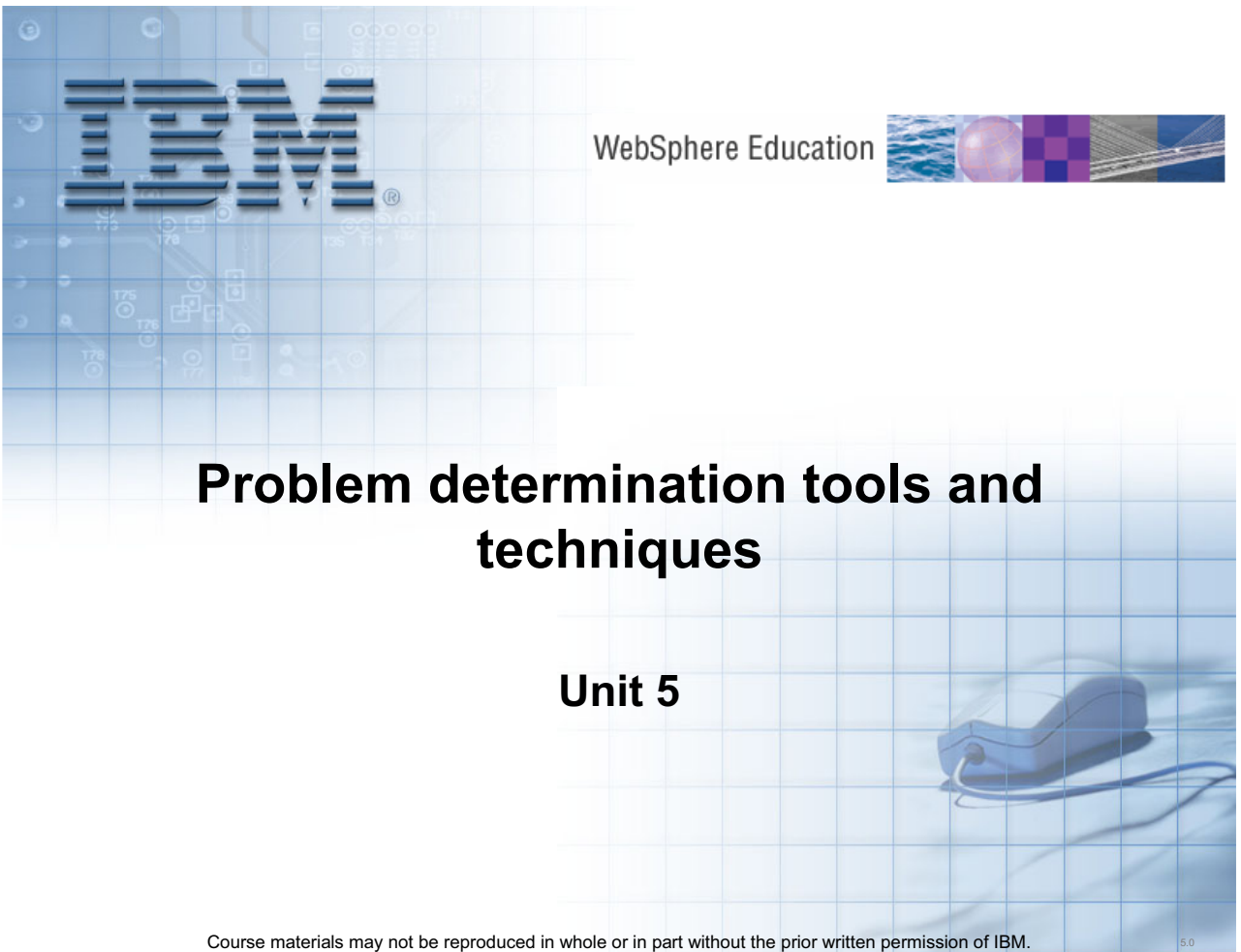


Figure 5-1. Problem determination tools and techniques

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Gather information for problem determination:
 - Search for information in the WebSphere Knowledge Base and the WebSphere Support page
 - Use product Information Centers
 - Use MustGather documents
 - Use the Troubleshooting Guide
- Build a detailed low-level timeline of events for deep analysis
- Identify and describe the main problem determination artifacts: logs, traces, dumps, PMI data, and so on
- Enable basic tracing of the server and HTTP plug-in
- Check product versions and patch levels
- Apply maintenance (APARs)
- Locate and interpret WebSphere FFDC logs
- Describe the types of tools used for problem determination
- List and compare some of the tools available
- Determine the appropriate tool for the problem
- Locate the tools needed

Figure 5-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose — To introduce the content of this unit.

Details —

Additional information —

Transition statement —

Topic objectives

After completing this topic, you should be able to:

- Identify resources for gathering information and researching a problem investigation
- Identify and locate common tools for troubleshooting

Figure 5-3. Topic objectives

WA5711.0

Notes:

Instructor notes:

Purpose — To introduce the content of this unit.

Details —

Additional information —

Transition statement —

WebSphere Knowledge Base search

- A good starting point for gathering information
- Composed of ISA, IGAA, developerWorks, product information centers, alphaWorks, and the IBM Support Web site
 - You can use ISA as your primary search engine
- Choose keywords to research the problem in available knowledge sources:
 - Error codes, exception names, APARs, fix packs, problem determination tools, MustGather documents, and so on
 - If there is no explicit error, use a high-level problem description
 - See list of problem categories on the IBM Support Web site
- Research on ISA or on the WebSphere Application Server Support Web page
 - Will automatically search product Information Center for error codes
 - WebSphere Application Server Support page:
 - <http://www.ibm.com/software/webservers/appserv/was/support/>

Figure 5-4. WebSphere Knowledge Base search

WA5711.0

Notes:

The next few slides relate to the information-gathering phase of a problem investigation.

All the problem determination data on the IBM Support site is organized into a set of predefined problem categories or “components.”

- Categories for WebSphere Application Server: “100% CPU Usage,” “Admin Console,” “Classloader,” “Crash,” and so on
- Also used to manage most support processes and other problem determination assets
- List of components and categories -
 - <http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599>

Library of technotes and other articles:

- Maintained by a specialized knowledge engineering team
- Includes known problems, APARs, common questions, troubleshooting instructions for many specific problems, problem determination tools, and so on

AlphaWorks provides links to new and emerging tools and technologies. Many problem determination tools can be found here - <http://www.alphaworks.ibm.com/>.

APAR: Authorized Program Analysis Report; tracks software defects reported by customers.

There are many ways to find help with troubleshooting WebSphere products. This article documents several resources -

http://www-128.ibm.com/developerWorks/websphere/techjournal/0701_supauth/0701_supauth.html.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM Support site search

WebSphere Application Server

Product support

Primary support resources

Flash 7 Sep, 2007: Compilation error within projects targeting a WebSphere Application Server V6.1

Flash 31 Aug, 2007: Take the guesswork out of finding support assistance

Flash 30 Aug, 2007: Interim Fix: IBM Java Cryptography Extension (JCE)
Expires on 18 May 2006

[\[View all Flashes\]](#)

Solve a problem

- [Troubleshooting guide](#)
- [Forums and newsgroups](#)
- [Featured documents](#)
- [IBM Support Assistant](#)
(tool to help resolve problems)
- [MustGather: Read first](#)
- [Technotes](#) | [APARs](#)

Download

- [Fixes by version](#)
 - [Feature packs by version](#)
 - [Recommended fixes](#)
 - [Latest Fix Packs](#)
- [\[View all downloads\]](#)

Learn

Search Support (this product)

Enter terms, error code or APAR #

Limit results (optional):

- [Solve a problem](#)
(Technotes, APARs)
- [Download](#)
(Fixes, Utilities, APARs)
- [Learn](#)
(Manuals, White Papers, etc.)

[→ Broaden or change scope](#)

[→ Tips for searching](#)

Figure 5-5. IBM Support site search

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM Software Support tool bar search

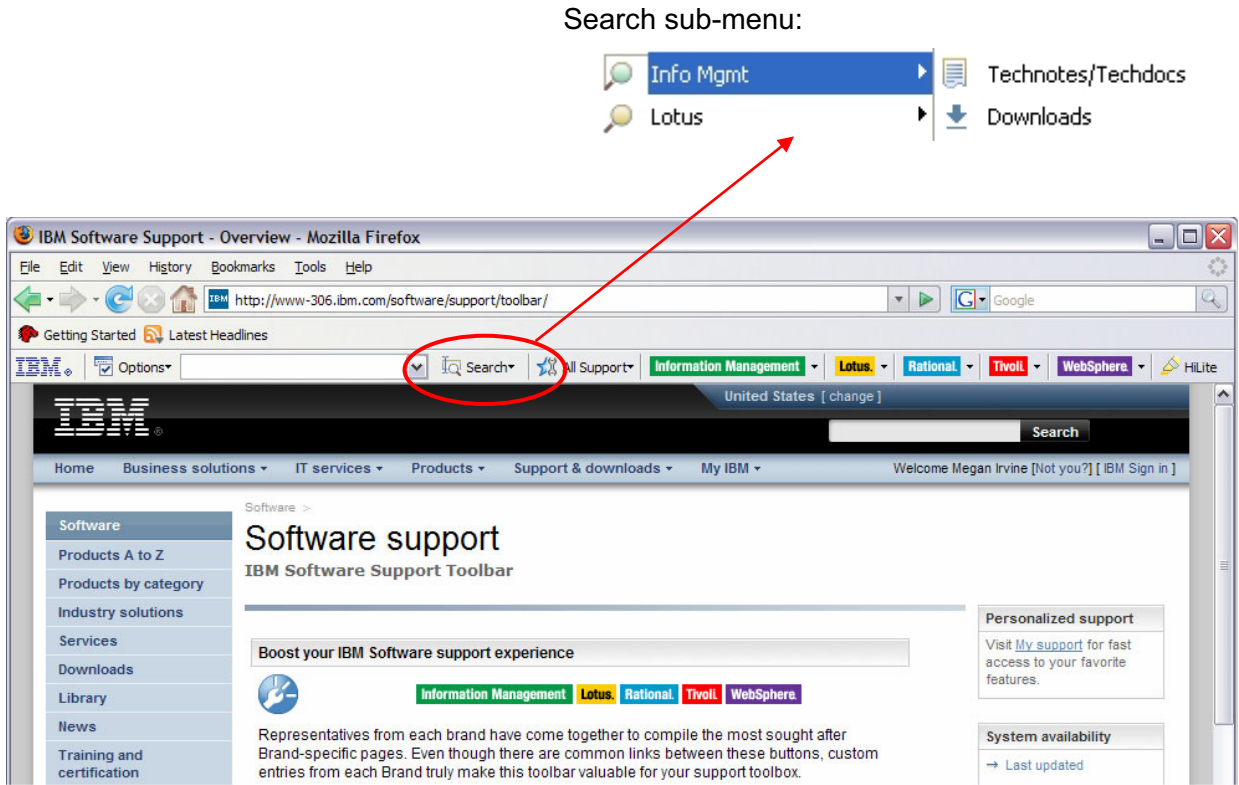


Figure 5-6. IBM Software Support tool bar search

WA5711.0

Notes:

Enter your search terms and press the Enter key, or click the Search button for more options. By default (just pressing the Enter key), your search will query multiple IBM Support resources.

If you need specific results for a specific brand, scroll to your brand and select from the submenu.

You can download the tool bar from <http://www.ibm.com/software/support/toolbar/>.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

MustGather example

TroubleShooting: Node agent and Deployment Manager discovery problems

Technote (FAQ)

Problem
 TroubleShooting for problems with the WebSphere® Application Server Node agent and Deployment Manager discovery component. This should help address common issues with this component before calling IBM support and save you time.

Solution

1. Learning more 2. Troubleshooting 3. Collecting data 4. Analyzing data

Steps to help resolve Node agent and Deployment Manager discovery problems

1. If you have multiple network interface cards (NIC) on the nodeagent or Deployment Manager node, and you do not want to use the primary NIC, you must apply [Interim Fix PQ71669](#) or use Fix Pack 1 or higher for version V5.0.

Collecting data tab -> MustGather document

Troubleshooting tab -> troubleshooting document

MustGather: Nodeagent and Deployment Manager discovery problems

Technote (FAQ)

Problem
 Collecting data for problems with the IBM® WebSphere® Application Server nodeagent and Deployment Manager discovery process component. Gathering this MustGather information before calling IBM support will help you understand the problem and save time analyzing the data.

Solution

1. Learning more 2. Troubleshooting 3. Collecting data 4. Analyzing data

If you have already contacted support, continue on to the component-specific MustGather information. Otherwise, click [MustGather: Read first for all WebSphere Application Server products](#).

Discovery component specific MustGather information
 In the administrative console under **System Administration > Nodes**, the node in question has a status of **Unknown**, or it is using **wsadmin** and is not able to get the `AdminContext1 MBean` object for the nodeagent. However, the nodeagent and the Deployment Manager are running.

The nodeagent and Deployment Manager must discover each other to have synchronization and administrative commands work properly on the base node. There are multiple reasons why the nodeagent and the Deployment Manager might not be able to talk to each other correctly.

1. Tracing the discovery process on the nodeagent and dmgr:
 - a. Stop all Application Server processes (no Java processes).
 - b. Modify the `seesrvr.xml` files for both nodeagent and dmgr:
 - i. Enable discovery tracing on the dmgr:
 1. Edit the `seesrvr.xml` file for the dmgr located in the following directory:


```
ND_PROFILES/config/cells/cellname/nodes/nodename/seesrvrs/dmgr
```

Figure 5-7. MustGather example

WA5711.0

Notes:

MustGather documents can be accessed from within ISA or on the IBM Support Web site.

The screen captures display a technote for node agent and deployment manager discovery problems.

On the IBM Support site, many MustGather documents now have companion “troubleshooting” and “analyzing data” documents (see for example - http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=troubleshooting&uid=swg21201625&loc=en_US&cs=utf-8&lang=en).

The troubleshooting document is what you check before you decide that you need to go through the MustGather document. The “analyzing data” document gives you pointers for how to interpret the info that you just collected from the MustGather document.

A majority of MustGather documents for WebSphere Application Server now have a corresponding AutoPD script (in ISA). So, you can either follow the steps from the MustGather document by hand, or run the AutoPD script, which will do the work more or less automatically.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM Guided Activity Assistant (IGAA)

- A new tool that brings together all three of these support elements: Information, tools, and processes
- Aims to help you answer:
 - What should you do next?
 - What diagnostic data should you analyze?
 - What tool should you use to analyze it?
 - How do you install and interact with that tool?
 - Other support-related process questions
- Accessed through ISA

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IGAA layout

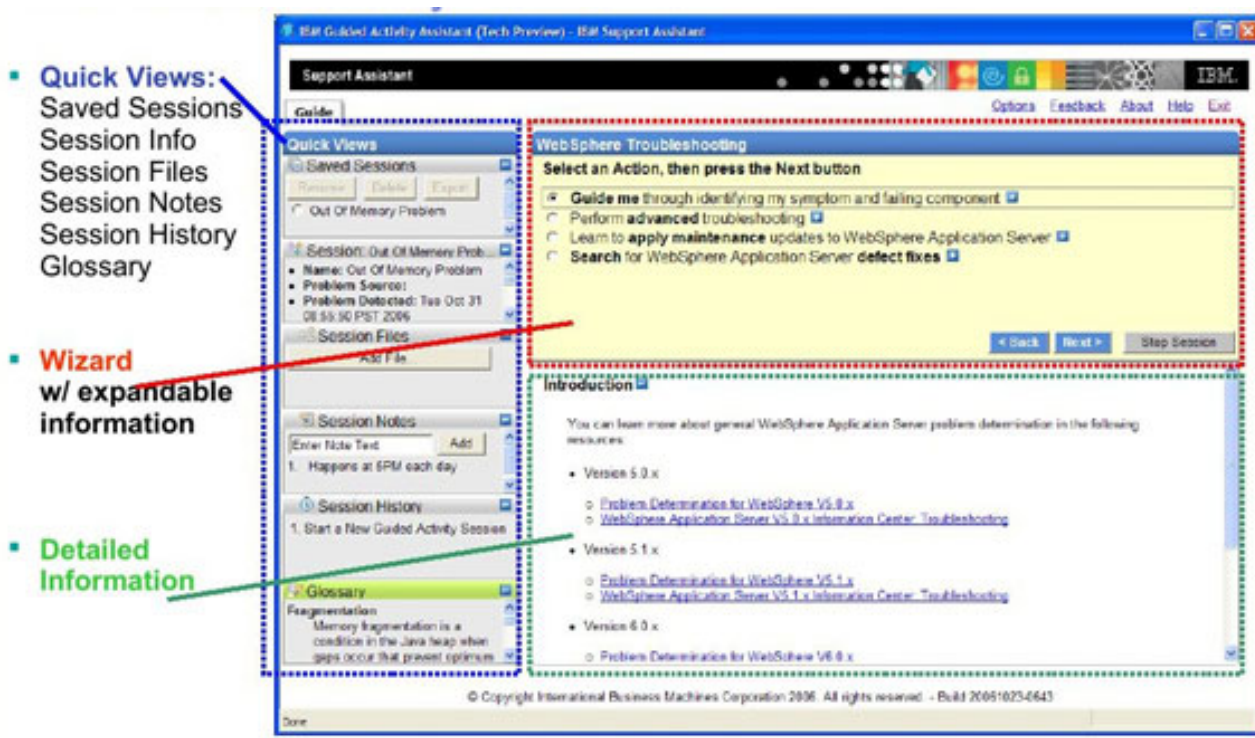


Figure 5-9. IGAA layout

WA5711.0

Notes:

The IGAA default layout consists of three primary sections:

Quick Views (blue section)

Quick Views are small expandable areas that provide session management and other capabilities for IGAA users. This section consists of:

- **Saved Sessions:** Resume, delete, import, and export sessions.
- **Session Information:** See the session metadata defined when creating the session.
- **Session Files:** Add and organize diagnostic data files collected during IGAA flows.
- **Session Notes:** Enter short notes describing observed session details, such as tool analysis results and session information.
- **Session History:** See the series of steps that have been taken for this session.
- **Glossary:** View the definition of common terms used in the IGAA flows.

You can customize which side of the panel you want the Quick Views section to appear (left or right) simply by dragging the Quick Views title bar from one side to the other.

Wizard

The Wizard section is the primary area for making path decisions as you progress through an IGAA flow. In addition, if there are specific steps or information that the content author determined to be particularly important, that information is presented in this section. The [+] symbols next to most path choices indicate that additional information is available to help you understand more about your path choices without reading the extensive details in the lower section.

Instructor notes:

Purpose —

Details — This section provides extra details about the current step that you are viewing. Additional text, links to existing resources (such as technotes, information centers, and Redbooks), and processing notes are all contained in this section. If you find you need extra information to help you work through an IGAA flow, look here for specific details.

Together, these three sections make up the interface with which you will work through IGAA flows. Take a look at a common scenario to illustrate how IGAA can help you identify and solve problems.

This article describes how to use IGAA -

http://www.ibm.com/developerworks/websphere/techjournal/0705_supauth/0705_supauth.html

Additional information —

Transition statement —

Types of tools and where to find them (1 of 4)

- Logging and tracing:
 - Troubleshooting panels in the administrative console
 - Log and Trace Analyzer (LTA) – part of Application Server Toolkit (AST)
 - Log Analyzer – accessible through ISA
 - Specialized tracing and runtime checks:
 - Connection leak detection
 - Session crossover detection
 - Enabled by tracing a specific component or setting a specialized custom property
- First failure data capture (FFDC)
 - Always enabled
 - Automatically captures key information when a potentially abnormal situation occurs
 - Data is collected in the `<profile_root>\logs\ffdc` directory
 - If an FFDC record is written, that does not necessarily mean that a serious problem occurred

Figure 5-10. Types of tools and where to find them (1 of 4)

WA5711.0

Notes:

The next few slides provide an overview of some of the types of tools used for troubleshooting and where they can be found. It is not an exhaustive list. Some of these tools will be discussed in more detail later on in the course. The following article provides a more detailed discussion of this topic -

http://www.ibm.com/developerworks/websphere/techjournal/0702_supauth/0702_supauth.html.

LTA is bundled with the following products:

IBM Build-to-Manage Toolkit

Available for download through IBM. This is a collection of technologies, tools, scenarios, and documentation designed for users to learn, adapt, and develop autonomic behavior in products and systems. All three types of the tooling are available in this package.

It is bundled as part of various IBM Rational, WebSphere (Application Server Toolkit), and Tivoli Provisioning Manager tooling. This is a stand-alone, Eclipse version. This version supports multi-events correlation.

IBM Open Process Automation Library, OPAL (for IBM Tivoli Enterprise Portal users)

IBM Open Process Automation Library, OPAL (for IBM Tivoli Enterprise Portal users) is a stand-alone Java Desktop version for problem isolation and triage to problem analysis. It supports single-event correlation and symptom analysis.

It can also be downloaded from here:

<http://www.ibm.com/developerworks/autonomic/btmpd/>

The first failure data capture (FFDC) log file saves information that is generated from a processing failure (for example, a Java exception).

- Captured data is saved in log files for use in analysis
- An index file that references all of the exceptions logged by FFDC
- An exception file for each exception type from each probe
- Capturing FFDC data does not affect performance

You can configure the number of days this information is saved (afterwards, it is deleted). Retrieve these log files using an FTP client from any other environment. Because the index and exception logs are text files, they can be viewed in any ASCII-capable text editor or viewer.

The FFDC configuration properties files are located in the properties directory under the WebSphere Application Server product installation. There are three properties files, but only the `ffdcRun.properties` file should be modified. You can set the `exceptionFileMaximumAge` property to configure the amount of days between purging the FFDC log files. The value of the `ExceptionFileMaximumAge` property must be a positive number.

The following redbook contains some good documentation on using the FFDC for problem determination - SG246880: *WebSphere for z/OS V5 Problem Determination*.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Types of tools and where to find them (2 of 4)

- Diagnostic providers:
 - A new feature in WebSphere Application Server V6.1 that allows you to query configuration and state information for a particular component
 - Accessible by using the administrative console
- Classloader viewer: New in V6.1; accessible through the administrative console
- JVM-level diagnostics (will be covered later in the course)
- Performance-related tools:
 - Performance Monitoring Infrastructure (PMI) – WebSphere-specific facility
 - Tivoli Performance Viewer (TPV), which is accessible through the administrative console is the primary tool for viewing PMI data
 - Request metrics – can be accessed by using Application Request Measurement (ARM) infrastructure
 - These tools can be useful in problem isolation and identification

Figure 5-11. Types of tools and where to find them (2 of 4)

WA5711.0

Notes:

Diagnostic providers:

- New feature in V6.1; an emergent technology
- With this technology, each WebSphere Application Server component (for example, Web container, connection manager, system management, or even components in the application) can be instrumented to export its own diagnostic provider object
- Each diagnostic provider object provides a uniform mechanism to perform three key functions:
 - Dump the configuration of a component.
 - Dump the state of a component.
 - Run self tests on a component.
- The following article on developerWorks provides a good introduction to diagnostic providers:

- http://www.ibm.com/developerworks/websphere/techjournal/0707_supauth/0707_supauth.html

The Java Diagnostics Guides also provide details about some of these tools and how to generate various types of dumps. You can find the V5 guide here:

<http://publib.boulder.ibm.com/infocenter/javasdk/v5r0/index.jsp>

Request metrics are available from many products, making it possible to follow a request from end to end in a complex system. Both PMI and request metrics are exported through public APIs, making it possible to write specialized or third-party tools to use this information.

Tivoli Composite Application Manager (TCAM) family of tools is the comprehensive platform for working with performance data, including PMI, request metrics, and other techniques.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Types of tools and where to find them (3 of 4)

- Monitoring and detection:
 - Hung thread detection facility – hooked into the WebSphere Application Server run time
 - Performance and diagnostic advisors – accessible through administrative console

- Specific sub-system investigation:
 - System Management Configuration Validation – can detect errors in the XML configuration files; accessible through the administrative console
 - DumpNameSpace – dumps the contents of the JNDI namespace at a particular server; shipped as a standalone product with WebSphere Application Server (*<install_root>/bin*)
 - Class Loader Viewer – can help in resolving class loading issues; accessible through Troubleshooting menu of administrative console

Figure 5-12. Types of tools and where to find them (3 of 4)

WA5711.0

Notes:

You can specify a hang threshold, or timeout period, for threads, and receive alerts about potentially hung threads.

The **System Management Configuration Validation** facility can perform automated checks to detect inconsistencies and errors in the complex set of XML files that contain the entire WebSphere Application Server system configuration. Such errors, though relatively rare in recent versions of the product thanks to many runtime safety checks, can still crop up due to as-yet-undiscovered product defects, unexpected events occurring during configuration operations (like crashes), or operator mistakes during configuration. This facility is embedded inside the WebSphere Application Server run time itself, and can be invoked from the administration console (in the Troubleshooting panel).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Types of tools and where to find them (4 of 4)

- Installation issues:
 - Installation verification tool (IVT) - verify each profile from its First steps console with the IVT tool (`<profile_root>/bin`)
 - Installation verification utility (installver) – does a more in-depth check of installed files for changes or inconsistencies; installed in `<install_root>/bin`
 - Update installer, VersionInfo, HistoryInfo, GenHistoryReport – used to track changes and apply fix packs; all bundled with WebSphere Application Server
- Debuggers and profilers – valuable to application support (but mostly outside the scope of this course)

Figure 5-13. Types of tools and where to find them (4 of 4)

WA5711.0

Notes:

Debuggers and profilers

The ability to debug and profile is also often very valuable to the application support process. WebSphere Application Server provides these facilities through the JVM, either through the JVMPi or JVMTI interfaces of that JVM. WebSphere Application Server system management makes it easy to set-up the appropriate JVM parameters to enable these facilities, either through the WebSphere Application Server administration console, or through a wsadmin script.

Rational and other Eclipse-based development tools, including **Rational Application Developer** and the **WebSphere Application Server Toolkit** include a powerful debugger and profiler tool that connects to these facilities. For profiling-related work, you might also consider using the **Performance Inspector** family of tools, which provides a variety of tools to extract and analyze runtime performance information from a JVM, using these same basic interfaces. These tools are available on alphaWorks (for Windows) or Sourceforge (for Linux).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tools available externally

- IBM Support Assistant (ISA)
 - Not installed with WebSphere Application Server
 - Available as a free download from here:
<http://www.ibm.com/software/support/isa/>
- In the meantime, until most tools are available through ISA, many tools can be found and downloaded individually from the Web
 - Search the Technotes on the IBM Support site (or in ISA), which contains references to individual tools
 - AlphaWorks – <http://www.alphaworks.ibm.com/>
- New tools become available on a regular basis; check the IBM Support site, ISA, developerWorks, and alphaWorks for the latest announcements

Figure 5-14. Tools available externally

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic objectives

After completing this unit, you should be able to:

- Describe and locate the different types of logs that can be used for problem determination
- Enable tracing
- View and interpret log and trace files

Figure 5-15. Topic objectives

WA5711.0

Notes:

Instructor notes:

Purpose — To introduce the content of this unit.

Details —

Additional information —

Transition statement —

Logs and tracing

- Logs:
 - Report key events in the system
 - Enabled by default
 - Incur minimal performance usage
 - Intended for users and administrators
- Tracing:
 - Application server code-level events; level of detail can be configured
 - Some tracing is on by default; more detailed tracing must be enabled by specifying a trace string
 - Intended for more technical users
- IBM service logs (activity.log):
 - Consolidates key messages on a particular node
 - Contains extended service information
- JMX- based monitoring:
 - Most key events are exported as JMX events
 - A variety of tools can be built for remotely monitoring and capturing logging information

Figure 5-16. Logs and tracing

WA5711.0

Notes:

The Tivoli Monitoring for Web Infrastructure tool uses JMX-based monitoring.

Default trace specification

In V6.1, the default trace string is `*=info`. This controls the logging level of the `SystemOut.log` file. If you only want errors to be logged, you can specify `*=fatal` or `*=severe`. For more messages, you can specify `*=detail`.

The settings `*=fine` through `*=all` produce a `trace.log` file.

Tracing is explained further later in this unit.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Log files and locations

- Log files:
 - SystemOut.log and SystemErr.log - Standard JVM output and error log
 - startServer.log and stopServer.log
 - Startup and shutdown of the application servers
 - activity.log - events that show a history of activities
 - Use Log Analyzer to read output from this file
 - trace.log – output from diagnostic trace
 - Destination and name are configurable
 - http_plugin.log – not in `<was_root>`
 - Location: `<plugin_root>\logs\<webserver_name>`
 - native_stdout.log and native_stderr.log
- The destination and names for the log files are configurable. The default location is `<was_root>\profiles\<profile_name>\logs\<server_name>`

Figure 5-17. Log files and locations

WA5711.0

Notes:

All WebSphere Application Server log files are stored in the `<was_root>\profiles\<profile_name>\logs` directory, by default.

SystemOut.log and SystemErr.log are the default names for the JVM logs. They contain server- as well as user-program information (sent by a System.out.xxx code in the program).

startServer.log and stopServer.log can also be found under the `<was_root>\logs\<servername>` directory, they contain information logged by the server as it starts up and shuts down. The IBM service log (activity.log) file size can be set by using the administrative console. You can also disable the activity.log.

Types of logs:

- **JVM logs:** Created by redirecting the System.out and System.err streams of the JVM to independent log files

- One set of JVM logs for each application server and all of its applications located by default in the `<install_root>/profiles/<profile_name>/logs/<server_name>` directory
- **Process logs:** Contain two output streams (stdout and stderr) that are accessible to native code running in the process
 - One set for each application server.
- **IBM service log:** Contains both the WebSphere Application Server messages that are written to the System.out stream and some special messages that contain extended service information that is normally not of interest, but can be important when analyzing problems.
- The HTTP server plug-in maintains a special log.

Java virtual machine (JVM) logs

The JVM logs are created by redirecting the System.out and System.err streams of the JVM to independent log files. WebSphere Application Server writes formatted messages to the System.out stream. In addition, applications and other code can write to these streams using the print() and println() methods defined by the streams.

In the case of a WebSphere Application Server Network Deployment configuration, JVM logs are also created for the deployment manager and each node agent because they also represent JVMs.

Process logs

WebSphere Application Server processes contain two output streams that are accessible to native code running in the process. These streams are the stdout and stderr streams. Native code, including Java virtual machines (JVM), might write data to these process streams. In addition, JVM provided System.out and System.err streams can be configured to write their data to these streams also.

As with JVM logs, there is a set of process logs for each application server, since each JVM is an operating system process, and in the case of a WebSphere Application Server Network Deployment configuration, a set of process logs for the deployment manager and each node agent.

IBM service logs

The IBM service log contains both the WebSphere Application Server messages that are written to the System.out stream and some special messages that contain extended service information that is normally not of interest, but can be important when analyzing problems. There is one service log for all WebSphere Application Server JVMs on a node, including all application servers. The IBM Service log is maintained in a binary format and requires a special tool to view. This viewer, the Log Analyzer, provides

additional diagnostic capabilities. In addition, the binary format provides capabilities that are utilized by IBM Support organizations.

The HTTP server plug-in log will be covered later in this presentation.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Viewing logs

- Most logs can be viewed with a text editor
- Runtime messages can be viewed on the administrative console (under Troubleshooting)
- Activity.log is written in binary format; use a tool such as Log Analyzer in ISA, or LTA to read it

Figure 5-18. Viewing logs

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Configuring JVM Logs

- Troubleshooting -> Logs and Trace -> <server_name> -> JVM Logs
- Alternative: Servers -> Application servers -> <server_name> -> Logging and Tracing -> JVM Logs
- System.out and System.err logs can be configured from this page
- Logs are self-managing
 - Can roll over based on time or file size
 - Number of historical log files is configurable
- To view logs through the console, use the **Runtime** tab

The screenshot shows the 'Runtime' configuration tab for 'System.out'. The 'File Name' is set to `${SERVER_LOG_ROOT}/SystemOut`. The 'File Formatting' is set to 'Basic (Compatible)'. Under 'Log File Rotation', 'File Size' is selected with a 'Maximum Size' of 3 MB and a 'Repeat Time' of 24 hours. 'Time' rotation is not selected. The 'Maximum Number of Historical Log Files' is set to 2. Under 'Installed Application Output', both 'Show application print statements' and 'Format print statements' are checked.

Figure 5-19. Configuring JVM Logs

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Viewing runtime messages in the console

The screenshot shows the WebSphere console interface. On the left, a navigation tree is visible with 'Runtime Messages' selected under the 'Troubleshooting' category. The main content area displays a 'Runtime Events' window with a table of events. The table has three columns: 'Timestamp', 'Message Originator', and 'Message'. The first message is circled in red.

Timestamp	Message Originator	Message
Apr 4, 2006 8:42:35 AM CDT	com.ibm.ws.security.registry.nt.NTLocalDomainRegistryImpl	SECJ0340E: Could not get the uniqueId for the group
Apr 4, 2006 8:42:35 AM CDT	com.ibm.ws.security.registry.nt.NTLocalDomainRegistryImpl	SECJ0350E: Could not get the uniqueId of the user
Apr 4, 2006 1:59:18 PM CDT	com.ibm.ws.performance.tuning.serverAlert.calc.DataAccessWrapper	Advisor caught exception. Advisor was unable to q
Apr 4, 2006 1:58:38 PM CDT	com.ibm.ws.performance.tuning.serverAlert.calc.DataAccessWrapper	Advisor caught exception. Advisor was unable to q
Apr 4, 2006 1:50:23 PM CDT	com.ibm.ws.performance.tuning.serverAlert.calc.DataAccessWrapper	Advisor caught exception. Advisor was unable to q
Apr 4, 2006 1:46:31 PM CDT	com.ibm.ws.webcontainer.WebContainer	SRVE0017W: A WebGroup/Virtual Host to handle /Xsea
Apr 3, 2006	com.ibm.ws.security.role.RoleBasedAuthorizerImpl	SECJ0306E: No

Figure 5-20. Viewing runtime messages in the console

WA5711.0

Notes:

While viewing runtime messages, first select the Error, Warning, or Information category links (a count of zero means nothing is available). The details for the selected category are shown. Selecting one of these links gives you more detailed information.

Note that you may have multiple pages of messages, the button on the bottom of the page will allow you to see them all.

In WebSphere Application Server V6.1, most runtime messages are designed with improved message text. Additional components have messages now.

Information is displayed on the detail screen for the event for you to resolve the problem with user action.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

HTTP plug-in logs and tracing

- Click **Servers -> Web Servers -> <web_server_name> Plug-in Properties -> Configuration tab -> Plug-in logging** to edit fields
- Default location:
`<plugins_root>/logs/<web_server_name>/http_plugin.log`
- Set the Log level to Trace to trace all the steps in the request process (Caution: This produces a lot of output)

Figure 5-21. HTTP plug-in logs and tracing

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Embedded HTTP Server logs

- Administrative Console panels for configuring embedded HTTP server logs (access and error)
- From main application server panel, click HTTP Error and NCSA Access Logging
- Access and error logs can be controlled separately
- When maximum file size is reached, oldest entries are pruned

The screenshot shows the 'Configuration' tab of the Embedded HTTP Server configuration. It is divided into three sections:

- General Properties:**
 - Enable service at server startup
- NCSA Access log:**
 - Enable access logging
 - * Access log file path: (Note: the image shows a partial path)
 - * Access log maximum size: MB
 - * NCSA access log format:
- Error log:**
 - Enable error logging
 - * Error log file path:
 - * Error log maximum size: MB
 - * Error log level:

Figure 5-22. Embedded HTTP Server logs

WA5711.0

Notes:

To enable the access or error log, you must check the “Enable service at server startup” check box, and also the check box for the specific log you want to enable. Notice there is no **Runtime** tab for these logs. Logging will only begin once you have saved these changes to your configuration and restarted the application server.

Click **Servers > Web Servers -> web_server_name Plug-in Properties -> Configuration tab -> Plug-in logging** to edit fields.

The default location is `<plugins_root>/logs/<web_server_name>/http_plugin.log`

Set the Log level to Trace to trace all the steps in the request process.



Important

Use caution when setting the Log level to Trace as this produces a lot of output.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tracing: Differences between V5.x and V6.x

- **=enabled**

- In WebSphere Application Server V5.x, all trace strings end in =enabled. For example:
 - com.ibm.ws.webservices.*=all=enabled
- In V6.x, this parameter was dropped, and so the same trace specification in V6.x would be:
 - com.ibm.ws.webservices.*=all

- **Default trace specification**

- In WebSphere Application Server V5.x, the default trace string is: *=all=disabled
- In WebSphere Application Server V6.x, the default trace string is *=info

Figure 5-23. Tracing: Differences between V5.x and V6.x

WA5711.0

Notes:

If a trace string containing =enabled is specified in V6.x, it will be changed to the new format automatically.

Default trace specification

In WebSphere Application Server V5.x, the default trace string is *=all=disabled. Therefore, even if tracing is enabled, no components are being traced so no trace.log will be created. At least one =enabled must be set for tracing to actually occur in WebSphere Application Server V5.x.

In WebSphere Application Server V6.x, the default trace string is *=info. The logging level of the SystemOut.log file can be controlled with this setting. For example, if you only want errors in the logs, you might use *=fatal or *=severe. However, if you want as much detail as possible, you might change this to *=detail. The settings *=fine through *=all will produce trace output to a trace.log file.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Traces (1 of 2)

- Trace files show the time and sequence of methods called by WebSphere Application Server base classes, and you can use these files to pinpoint the failure
- Trace can be started
 - While server is running using Runtime Diagnostic Trace
 - When server is started using Configuration Diagnostic Trace
- Trace output can be directed to:
 - Memory ring buffer - dumped after trace stops
 - File
- Trace has a significant impact on performance
 - Enable temporarily for problem determination
 - Trace to file is slower than trace to memory ring buffer **Runtime tab**

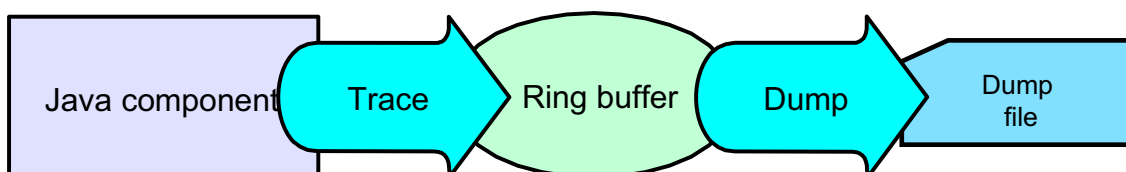


Figure 5-24. Traces (1 of 2)

WA5711.0

Notes:

Trace output allows administrators to examine processes in the application server and diagnose various issues.

On an application server, trace output can be directed either to a file or to an in-memory circular buffer. If trace output is directed to the in-memory circular buffer, it must be dumped to a file before it can be viewed.

On an application client or stand-alone process, trace output can be directed either to a file or to the process console window.

In all cases, trace output is generated as plain text in either basic, advanced or log analyzer format as specified by the user. The basic and advanced formats for trace output are similar to the basic and advanced formats that are available for the JVM message logs.

The procedure for using trace is as follows:

1. Configure an output destination to which trace data is sent.
2. Enable trace for the appropriate WebSphere Application Server or application components.

3. Run the application or operation to generate the trace data.
 4. Analyze the trace data or forward it to the appropriate organization for analysis.
- Click **Troubleshooting -> Logs and Trace -> *server_name*** in the administrative console.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Enabling trace

- **Troubleshooting -> Logs and Trace -> <server_name> -> Diagnostic Trace**
- **Enable Log** checkbox enables tracing
- On **Configuration** tab, server restart is required for settings to take effect
- **Configurable output**
 - Memory buffer or file

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

Trace Output Format
Basic (Compatible)

Apply OK Reset Cancel

Figure 5-25. Enabling trace

WA5711.0

Notes:

Specifying the trace settings under the **Configuration** tab will require a server restart for the tracing to take effect. However, enabling the tracing under the **Runtime** tab will take effect immediately (which can be very useful for tracing environments such as a production environment, which can rarely afford a server restart). However, it is important to know that after a server restart, the runtime settings are lost unless the **Save runtime changes to configuration as well** box is checked. Also, runtime traces only report the current system information; if the server is already having issues, enabling runtime traces will not report what caused the problem to begin, only why current actions fail. Therefore, IBM Support recommends that the trace settings be made under the **Configuration** tab so that server startup can be reviewed and the root cause of the problem can be determined.

The Maximum File Size and Maximum Number of Historical Files are important to ensure the trace information being captured does not get wrapped and lost. By default, these are set to 20 MB for the file size, and one backup file. This means that when the trace.log file reaches 20 MB in size, a new trace.log will get created and the previous trace.log will be renamed to trace_<date>.log where <date> is the date and time of the last entry in the trace. However, once the new trace.log reaches the 20 MB in size again, it will then wrap

into a new trace file as before, but the previous trace_<date>.log will be deleted. Therefore, it is important to ensure the Number of Historical Files and Maximum File Size is large enough when recreating a complex or time consuming problem. The largest value that can be inserted for the Historical Files is 20. The WebSphere Application Server support team prefers to keep the Maximum File Size no larger than 50 MB, as the traces become more difficult to load and read if the file size is too large.

The Diagnostic Trace Service box looks mostly the same as it did in previous versions. The **Configuration** and **Runtime** tabs behave as they always have, with **Configuration** affecting the configuration repository and taking effect at the next startup, while **Runtime** takes effect immediately but is only optionally persisted to the server configuration. The “Enable Log” check box is enabled per default. Because the Log Detail Level is set to `*=info`, there is no trace output. The major change on this panel is the absence of a space to enter the trace string. Trace strings have been moved to a separate panel (Log Detail Level).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Trace dump and run time

- On **Runtime** tab, settings take effect immediately (useful in production environments)
- View and dump available in the **Runtime** tab of diagnostic trace
- LTA can be used to analyze trace output, but you may prefer to use your favorite editor
- Before you can view or dump trace you need to specify log detail level

The screenshot shows a configuration dialog box with two tabs: 'Configuration' and 'Runtime'. The 'Runtime' tab is active. Under the 'General Properties' section, there is a checkbox for 'Save runtime changes to configuration as well'. Below this is the 'Trace Output' section, which has two radio button options: 'Memory Buffer' and 'File'. The 'File' option is selected. Under 'Memory Buffer', there is a 'Maximum Buffer Size' field set to an empty box followed by 'thousand entries' and a 'Dump File Name' field. A 'Dump' button is located below these fields. Under 'File', there is a 'Maximum File Size' field set to '20' followed by 'MB', a 'Maximum Number of Historical Files' field set to '1', and a 'File Name' field containing the path '\${SERVER_LOG_ROOT}/trace.log'. A 'View' button is located below the file name field. At the bottom of the dialog are four buttons: 'Apply', 'OK', 'Reset', and 'Cancel'.

Figure 5-26. Trace dump and run time

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Setting the Log Detail Level

- **Logs and Tracing -> Change Log Detail Level**
- Log detail level affects tracing *and* regular logging
 - Setting levels below ***=info** reduces the amount of data in logs
 - ***=fatal** and ***=severe** log only error messages
 - ***=detail** creates a lot of output
- Trace levels (**fine**, **finer**, **finest**) appear in trace.log, when enabled
- Trace string can be typed in or set using the graphical menu
- User-created applications can be instrumented and included in the trace output

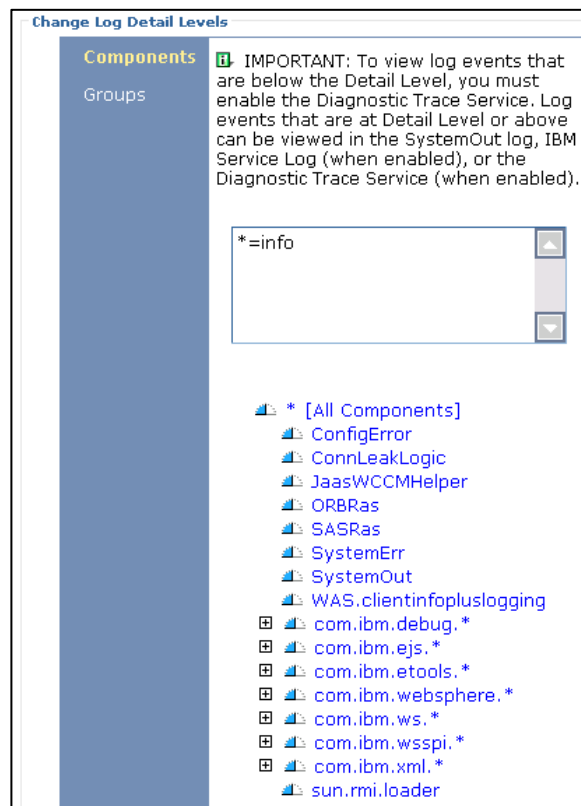


Figure 5-27. Setting the Log Detail Level

WA5711.0

Notes:

Log Levels control which events are processed by Java logging.

WebSphere Application Server controls the levels of all loggers in the system. The level value is set from configuration data when the logger is created and can be changed at run time from the administrative console.



Note

Trace information, which are events at levels Fine, Finer and Finest, can only be written to the trace log. Therefore, if you do not enable diagnostic trace, setting the log detail level to Fine, Finer or Finest will not have an effect on the data that is logged.

Log String Syntax: <component / group> = <log level>

Examples:

```
com.ibm.ws.classloader.ClassGraph=finest
```

Enables finest trace level for com.ibm.ws.classloader.ClassGraph

```
EJBContainer=fine
```

Enables least verbose trace level for all components in the EJBContainer group

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

Enables detailed trace for all classes in the com.ibm.ws.classloader package

```
*=info
```

Sets the log level for all components to info (default – no trace output)

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Enabling trace by using wsadmin

- Use these commands to enable tracing on the configuration (this example sets the trace string com.ibm.ws.*=all=enabled):

```
set server [$AdminConfig getid
  /Cell:<mycell>/Node:<mynode>/Server:<myserver>/]
set tc [$AdminConfig list TraceService $server]
$AdminConfig modify $tc {{startupTraceSpecification
  com.ibm.ws.*=all=enabled}}
$AdminConfig save
```

- To enable run time traces using wsadmin, use these commands:

```
set ts [$AdminControl queryNames
  type=TraceService,node=<mynode>,process=<myserver>,*]
$AdminControl setAttribute $ts traceSpecification
  com.ibm.ws.*=all=enabled
```

Figure 5-28. Enabling trace by using wsadmin

WA5711.0

Notes:

If access to the administrative console is not possible, tracing can also be enabled using the wsadmin utility. The information center contains more detailed instructions on enabling tracing using the wsadmin tool.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Trace header

- The header at the top of the SystemOut.log and trace.log files contain valuable information about the server that produced the log

```

***** Start Display Current Environment *****
WebSphere Platform 6.0 [ND 6.0.2.15 cf150636.04] running with process
name carterfinleyNode02Cell\carterfinleyNode01\server1 and process id
2744
Host Operating System is Windows XP, version 5.1
Java version = J2RE 1.4.2 IBM Windows 32 build cn142ifx-20061121 (ifix
112270: SR6 + 111682 + 111872 + 110979) (JIT enabled: jitc), Java
Compiler = jitc, Java VM name = Classic VM
was.install.root = C:\WebSphere60
user.install.root = C:\WebSphere60\profiles\AppSrv01
Java Home = C:\WebSphere60\java\jre
ws.ext.dirs = C:\WebSphere60\java\lib;C:\WebSphere60\profiles\AppSrv01\classes;C:\Web
Sphere60\classes;C:\WebSphere60\lib;C:\WebSphere60\installedChannels;C:
\WebSphere60\lib\ext;C:\WebSphere60\web\help;C:\WebSphere60\deploytool\
itp\plugins\com.ibm.etools.ejbdeploy/runtime
Classpath = C:\WebSphere60\profiles\AppSrv01\properties;C:\WebSphere60\properties;C
:\WebSphere60\lib\bootstrap.jar;C:\WebSphere60\lib\j2ee.jar;C:\WebSpher
e60\lib\lmpoxy.jar;C:\WebSphere60\lib\urlprotocols.jar
Java Library path = C:\WebSphere60\java\bin;. ;C:\WINNT\system32;C:\WINNT;C:\WebSphere60
\bin;C:\WebSphere60\java\bin;C:\WebSphere60\java\jre\bin;C:\WebSphere51\IBM
\WebSphere_MQ\Java\lib;C:\PROGRAM FILES\THINKPAD\UTILITIES;C:\WINNT\sys
tem32;C:\WINNT;C:\WINNT\System32\Wbem;(etc...)
Current trace specification = *=info:com.ibm.ws.security.*=all
***** End Display Current Environment *****

```

Figure 5-29. Trace header

WA5711.0

Notes:

This header is also produced each time the server is started. From this header, you can see the exact WebSphere Application Server level and Java version, the cell name, node name, and server name, along with the process ID.

Additionally, the contents of the class path and Java lib path are visible, as well as the WebSphere Application Server variables was.install.root and user.install.root. From a tracing perspective, the most useful part of the header is the last line, which shows you the current trace string being used:

Current trace specification = *=info:com.ibm.ws.security.*=all

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Default trace format

- **Basic format**-Trace events displayed in basic format use the following format:

```
[timestamp] <threadId> <className> <eventType> <methodName> <textmessage>
```

- Here is an example:

```
[2/5/07 13:13:49:457 EST] 0000000a ConfigFile > buildFileEntry() Entry
[2/5/07 13:13:49:457 EST] 0000000a ConfigFile 3 JAAS login configuration
file:
C:\WebSphere60\profiles\AppSrv01/properties/wsjaas.conf singleLogInFile: false
[2/5/07 13:13:49:497 EST] 0000000a ConfigFile 3 JAAS login configuration
file:
C:\WebSphere60\profiles\AppSrv01/properties/wsjaas.conf processed
[2/5/07 13:13:49:497 EST] 0000000a ConfigFile < buildFileEntry() Exit
```

Figure 5-30. Default trace format

WA5711.0

Notes:

The "Basic" trace format is the preferred trace format of the WebSphere Application Server support team, and it is also the default trace format.

A full listing of each of the Diagnostic Trace Service settings and their descriptions are documented in the WebSphere Application Server information center.

From the sample trace above, you can see:

- The time frame is on Feb 5, 2007 at 13:13:49 EST time.
- The trace output shows a single threaded process with ID 000000a.
- The class name executing is "ConfigFile".

If the class name is over 13 characters, only the first 13 characters will display.

Here is the breakdown of each trace entry line and the information it provides:

```
[2/5/07 13:13:49:457 EST] 0000000a ConfigFile > buildFileEntry() Entry
```

The **ConfigFile.buildFileEntry()** method begins execution here.

```
[2/5/07 13:13:49:457 EST] 0000000a ConfigFile 3 JAAS login configuration
file:
C:\WebSphere60\profiles\AppSrv01/properties/wsjaas.conf singleLogInFile:
false
```

A debug statement in the **buildFileEntry()** code.

```
[2/5/07 13:13:49:497 EST] 0000000a ConfigFile 3 JAAS login configuration
file:
C:\WebSphere60\profiles\AppSrv01/properties/wsjaas.conf processed
```

A debug statement in the **buildFileEntry()** code.

```
[2/5/07 13:13:49:497 EST] 0000000a ConfigFile < buildFileEntry() Exit
```

The **ConfigFile.buildFileEntry()** method has finished execution and exiting.

Not all WebSphere Application Server code outputs the method entry and exit points, and debug messages.

Instructor notes:

Purpose — Point out the different information fields for each trace format.

Details — Caution the students that trace log files can grow very large very quickly and that they should be careful to manage the space on their disk partitions.

Additional information —

Transition statement —

Reading a trace file

- Timestamps give good clues:
 - Time stamps are real system time values
 - Good when comparing traces from different processes
- Look for exceptions (search for exception from top)
 - Events prior to exception are probable causes
 - Events after exception are recovery attempts
- Often useful to follow a single thread

>	Entry to a method (debug)
<	Exit a method (debug)
A	Audit
W	Warning
X	Error
E	Event (debug)
D	Detail (debug)
T	Terminate (exits process)
F	Fatal (exits process)
I	Information
O	Program output
C	Configuration

- Log and Trace File Format:

[06/06/06 9:51:15:081 GMT] 3c07adad PMImpl A PMON0001A:PMI enabled

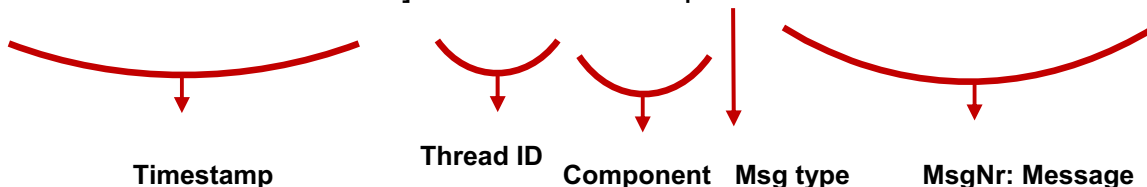


Figure 5-31. Reading a trace file

WA5711.0

Notes:

The following article gives a good introduction to interpreting trace data - http://www.ibm.com/developerworks/websphere/techjournal/0704_supauth/0704_supauth.html

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tools for viewing trace output: Trace Analyzer for WebSphere

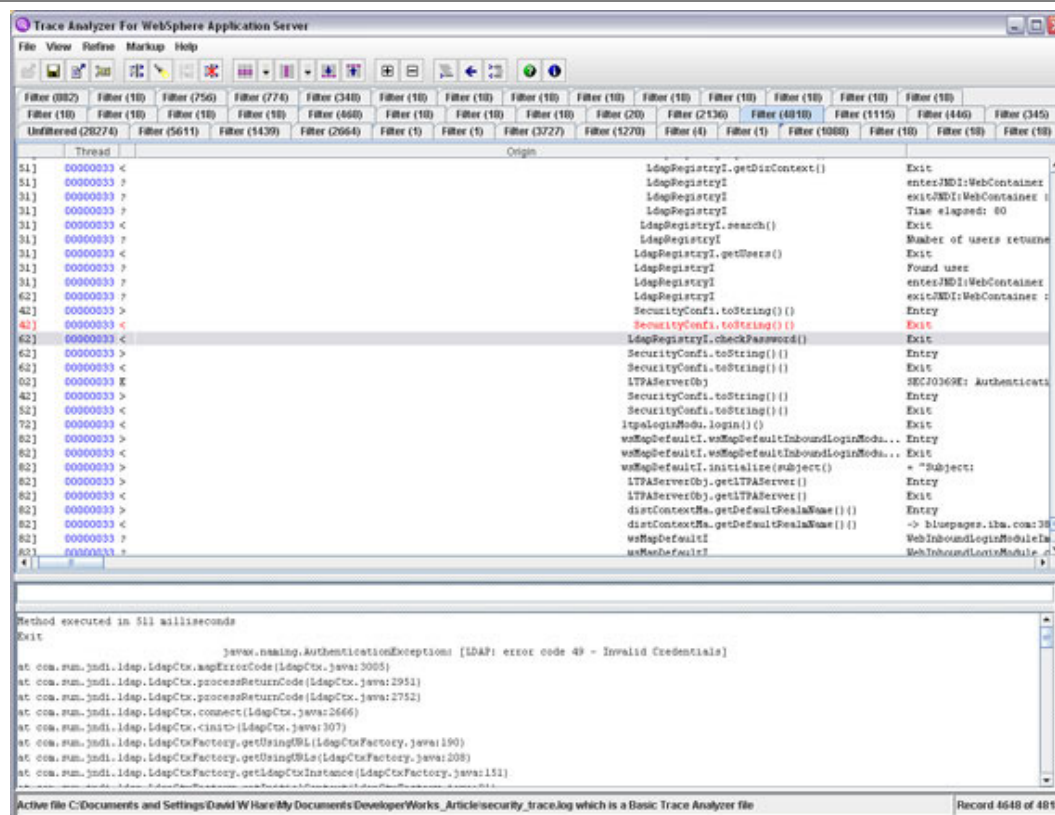


Figure 5-32. Tools for viewing trace output: Trace Analyzer for WebSphere

WA5711.0

Notes:

The Trace Analyzer for WebSphere Application Server enables you to view a complex text-based trace file in a more user-friendly GUI interface. The contents of the trace are divided into rows and columns, which makes the trace easier to read. The tool contains many useful features, such as finding the corresponding entry and exit point, following a single thread, and even generating a call stack.

When debugging a trace, it is often easier to pull the thread you are interested in so you only need to view trace entries associated for the respective thread. This can be accomplished with a script similar to pullThread.sh script, which can be found at http://www.ibm.com/developerworks/websphere/techjournal/0704_supauth/0704_supauth.html. By default, this script will print the output to the prompt, but you can easily pipe this out to create another log file.

The command to use this script is:

```
./pullThread.sh <threadID> <tracefile>
```

The command with actual parameter values looks like this:

```
./pullThread.sh 00000033 trace.log > NewTrace.log
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

For more information

- Articles on developerWorks:
 - Interpreting a WebSphere Application Server trace file
 - http://www.ibm.com/developerworks/websphere/techjournal/0704_supauth/0704_supauth.html
 - Features and tools for practical troubleshooting
 - http://www.ibm.com/developerworks/websphere/techjournal/0702_supauth/0702_supauth.html

Figure 5-33. For more information

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. Name some resources that you can use to gather information on how to troubleshoot a problem.
2. What is a MustGather document?
3. What is the default location for the WebSphere Application Server logs?
4. What is the default location for the HTTP plug-in logs?
5. True or false: Tracing cannot be started while the server is running.

Figure 5-34. Checkpoint

WA5711.0

Notes:

Write down your answers here:

- 1.
- 2.
- 3.
- 4.
- 5.

Instructor notes:

Purpose —

Details — There are several right answers for question 1. The answers page only lists a few of the main ones.

Additional information —

Transition statement —

Checkpoint

6. Trace output can be directed to _____ or _____.
7. Name some troubleshooting tools that are integrated into the WebSphere administrative console.
8. What tools are integrated into ISA?
9. True or false: Application Server Toolkit (AST) can be used to analyze logs and trace.

Figure 5-35. Checkpoint

WA5711.0

Notes:

Write down your answers here:

- 6.
- 7.
- 8.
- 9.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

1. Name some resources that you can use to gather information on how to troubleshoot a problem.
 - **IBM Support Assistant (ISA)**
 - **The WebSphere Support site**
 - **The Information Centers**
 - **And more**

2. What is a MustGather document?
 - **A MustGather document tells you the specific information that you need to collect for a particular type of problem.**

3. What is the default location for the WebSphere Application Server logs?
 - **<was_root>\profiles\<profile_name>\logs**

4. What is the default location for the HTTP plug-in logs?
 - **<plugins_root>/logs/<web_server_name>/http_plugin.log**

5. True or false: Tracing cannot be started while the server is running.
 - **False.**

Figure 5-36. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

6. Trace output can be directed to _____ or _____.
 - **Trace output can be directed to a file or ring buffer.**

7. Name some troubleshooting tools that are integrated into the WebSphere administrative console.
 - **Tivoli Performance Viewer and Performance Adviser**
 - **Classloader viewer**
 - **Configuration validator**

8. What tools are integrated into ISA?
 - **IBM Guided Activity Assistant (IGAA)**
 - **Collector**
 - **MDD4J**
 - **Log Analyzer**

9. True or false: Application Server Toolkit (AST) can be used to analyze logs and trace.
 - **True**

Figure 5-37. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Gather information for problem determination:
 - Search for information in the WebSphere Knowledge Base and the WebSphere Support page
 - Use product Information Centers
 - Use MustGather documents
 - Use the Troubleshooting Guide
- Build a detailed low-level timeline of events for deep analysis
- Identify and describe the main problem determination artifacts: logs, traces, dumps, PMI data, and so on
- Enable basic tracing of the server and HTTP plug-in
- Check product versions and patch levels
- Apply maintenance (APARs)
- Locate and interpret WebSphere FFDC logs
- Describe the types of tools used for problem determination
- List and compare some of the tools available
- Determine the appropriate tool for the problem
- Locate the tools needed

Figure 5-38. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise objectives

After completing this exercise, you should be able to:

- Use the administrative console to view messages
- View logs for an application server
- Use AST to view service and JVM logs
- Enable tracing on an application server
- View FFDC log file

Figure 5-39. Exercise objectives

WA5711.0

Notes:

Instructor notes:

Purpose — To introduce the content of this unit.

Details —

Additional information —

Transition statement —

Exercise

- Exercise 2: Introduction to problem determination tools

Figure 5-40. Exercise

WA5711.0

Notes:

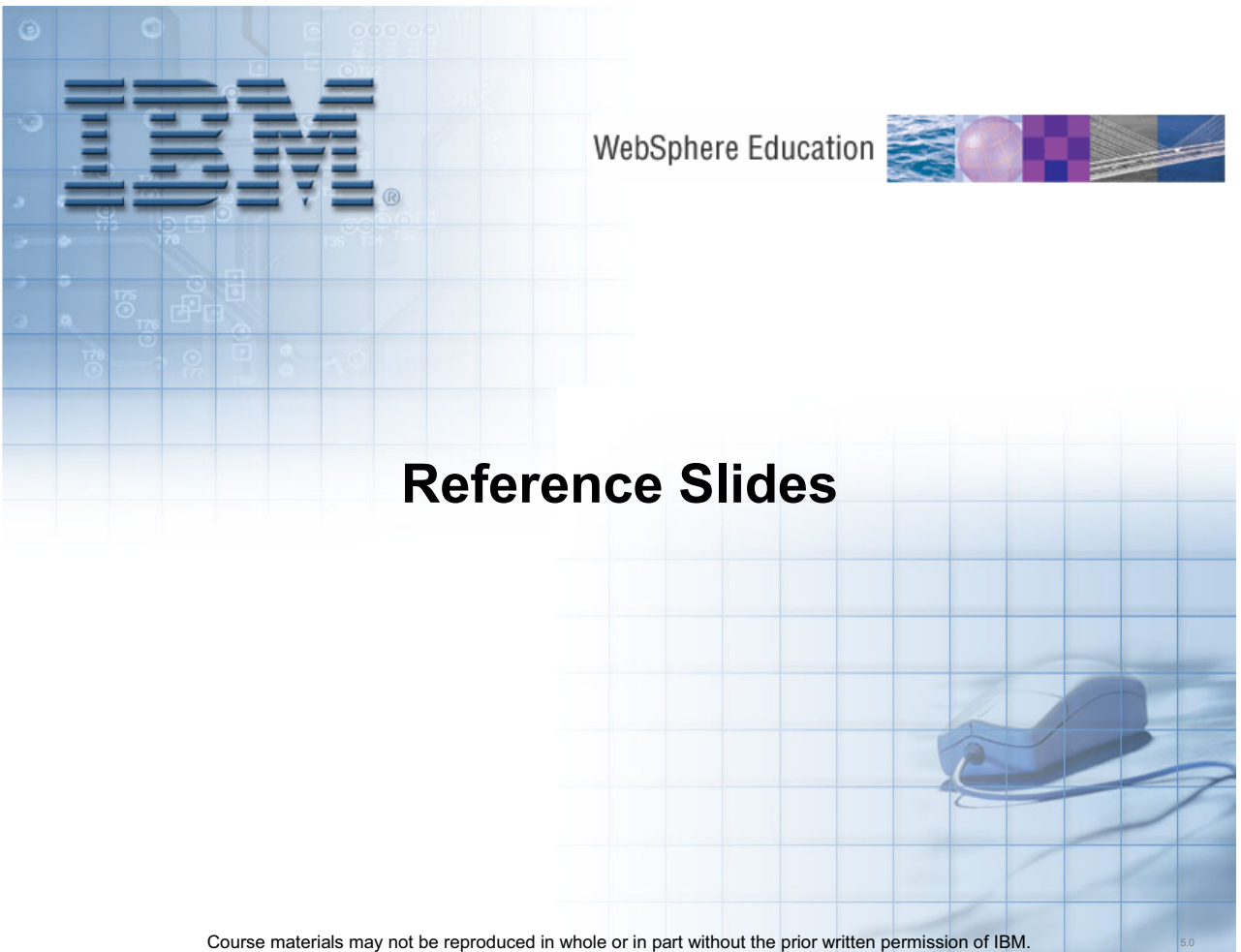
Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.0

Figure 5-41. Reference Slides

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Log and Trace Analyzer for Java Desktop (LTA-JD)

Use to view and analyze log and trace files.

Compares log or trace data against a symptom database.



Figure 5-42. Log and Trace Analyzer for Java Desktop (LTA-JD)

WA5711.0

Notes:

The LTA-JD application consists of the following main capabilities:

- An event normalization module in which various kinds of information are collected and transformed into a format understandable by the system
- An event filtering and visualization module in which events are collected from a managed resource and displayed to system administrators and support personnel
- A simple symptom definition module in which symptoms are composed and associated with visualization parameters
- An integrated symptom visualization module that presents the symptoms by overlaying their visualization aspects with those of normal events (which are in turn components of a symptom)
- A dynamic symptom avoidance module in which symptoms trends are detected and recommendations are suggested to human administrators on what to do to avoid the symptom manifesting itself (thus avoiding the problem before it happens)

The Log and Trace Analyzer in AST also provides stand-alone and plug-in support for new and existing users of the JSR-047 Java Logging API and a lot of common loggings (DB2, AIX, all trace and log files of WebSphere Application Server, Portal Server, and so on). The IBM Agent Controller allows remote operation of this tool.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Java Logging Architecture - Overview

- WebSphere V6 uses a standard Java logging package, `java.util.logging`
- Provides a portable and extensible logging API

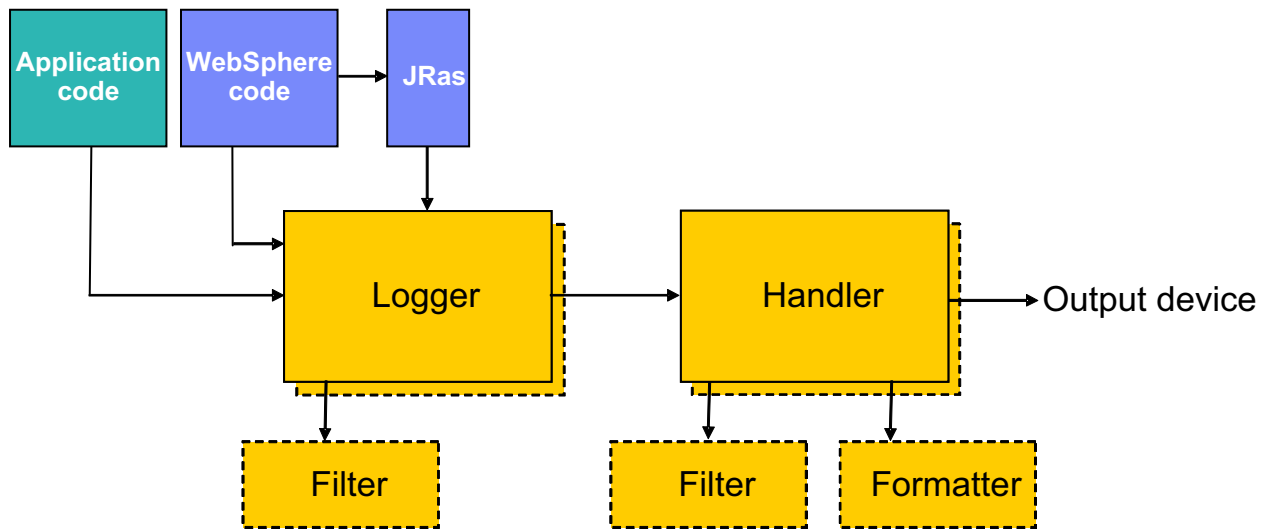


Figure 5-43. Java Logging Architecture - Overview

WA5711.0

Notes:

Both application code and WebSphere code make use of Logger objects to put data onto the logstream, in the form of LogRecord objects.

Logger objects can be associated with one or more Handler objects.

Handlers represent output devices. For instance, one handler would represent the service log, while another might represent the StandardOut log.

Filters are used to decide which messages get forwarded through the stream and which do not. For instance, A filter would be attached to the StandardOut Handler such that only messages intended for StandardOut could pass through. You could also exclude messages that contained a particular key using a filter.

Formatters are used by Handlers to format log data for output. Localization could be implemented using a Formatter, for example.

JRas is integrated with the Java logging API, so that Loggers and Handlers can receive and process all messages, regardless of whether they were logged using JRas or Java logging.

Instructor notes:**Purpose —**

Details — This chart is mainly targeted at programmers and people who intended to actually modify the configuration of the logging system. However, 99% of users and people who do problem determination do not actually change the logging configuration—they just need to know how it is set up by default on their system, where to find things and how to turn on trace.

You may want to de-emphasize this chart, or approach it from two different angles:

- How to find and use the logs and traces
- How to configure and customize your own logs and traces

Additional information — This relates to JSR47.

Transition statement —

Diagnostic Provider infrastructure

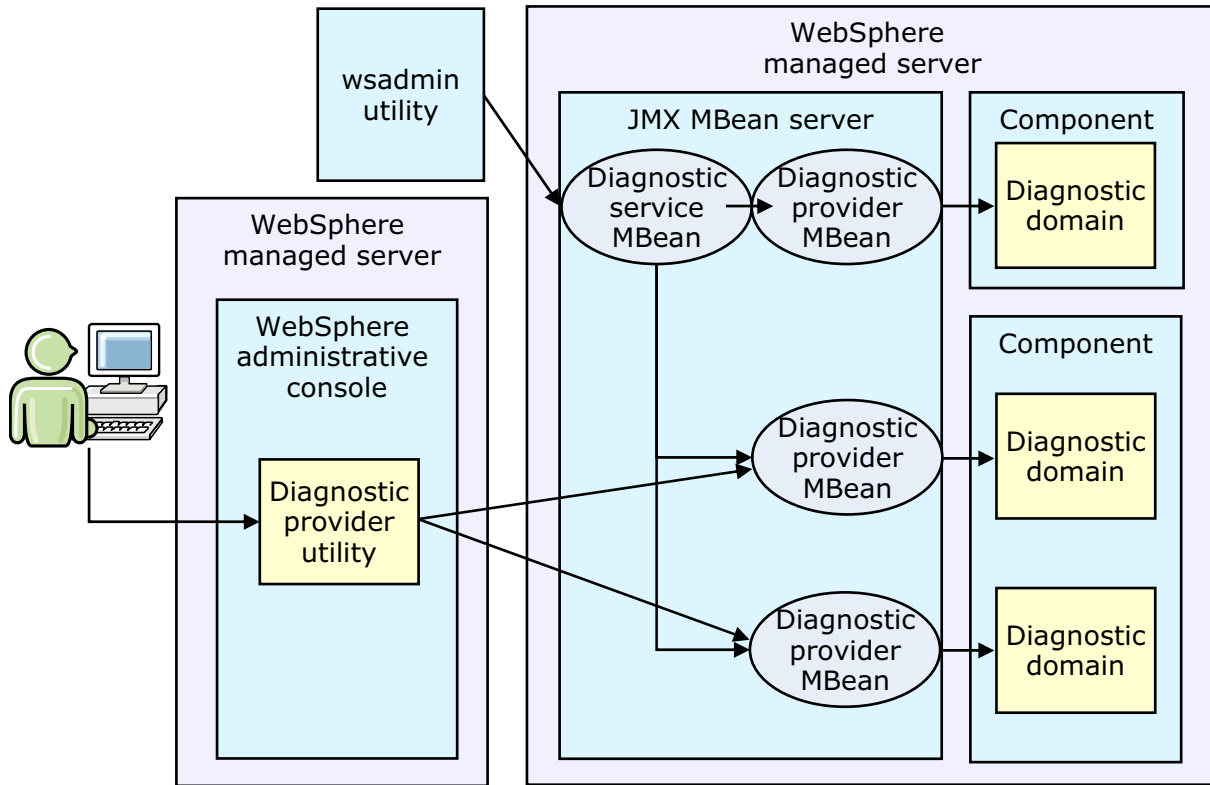


Figure 5-44. Diagnostic Provider infrastructure

WA5711.0

Notes:

This image shows the relationships between the parts that make up the Diagnostic Provider (DP) utility.

A single diagnostic domain receives its diagnostic services from a Diagnostic Provider MBean. The Diagnostic Provider MBean enables you to query the startup configuration, current configuration, and current state of the diagnostic domain. In addition, Diagnostic Provider MBeans can also provide access to any self diagnostic tests that are available from the diagnostic domain. Some characteristics of Diagnostic Provider MBeans include:

- Diagnostic Provider MBeans are Java Management Extensions (JMX) MBeans.
- Diagnostic Provider MBeans all implement a DiagnosticProvider interface which includes methods for configuration dumps, state dumps, and self diagnostic tests.
- Diagnostic Provider MBeans provide a way to expose information about running components so administrators can more easily debug problems related to those components. As with other MBeans running in WebSphere Application Server, they can be accessed from JMX client code, or through the wsadmin tool.

Diagnostic Provider MBeans are efficient at delivering Java object representations of configuration, state, and self test information. This is good for when programs interact. For human users to access the information, WebSphere Application Server provides a set of facilities to extend the value of Diagnostic Provider MBeans.

The Diagnostic Service MBean provides methods to convert Diagnostic Provider MBean output into human readable formats. The Diagnostic Service MBean also provides some methods to facilitate looking up the Diagnostic Provider MBeans on the same server as the Diagnostic Service MBean. For administrators that want to access diagnostic data from a command line, the wsadmin tool can be used directly with the Diagnostic Service MBean to get formatted results.

The Diagnostic Provider utility is a set of panels included in the WebSphere Application Server administration console through which administrators can interact with Diagnostic Provider MBeans. The Diagnostic Provider utility is a simple front end in the administration console that presents the available set of Diagnostic Provider MBeans present on each managed server, and provides a means to execute and view the results of configuration dumps, state dumps, and diagnostic self tests.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Example Diagnostic Provider scenario

1. A log entry that contains a Diagnostic Provider ID (DPID)* indicates that something has gone wrong in a specific component
2. The system administrator sees the log entry through the runtime messages panel
3. The administrator clicks a button on the runtime message panel to start a state dump or a configuration dump, or to be taken to the list of component self tests
4. From the self test, the administrator is warned that the component is configured in a way that might lead to poor performance or failures

*In this case, the DPID is registered with the logger of the component

Figure 5-45. Example Diagnostic Provider scenario

WA5711.0

Notes:

When the administrator works with a component with a Diagnostic Provider, and the Diagnostic Provider ID is *not* registered with the logger of the component, the situation might unfold like this:

1. A log entry which does not contain a DPID indicates that something has gone wrong in a component.
2. The system administrator sees the log entry through the runtime messages panel.
3. The system administrator uses the administrative console to navigate through the available set of Diagnostic Providers and selects one that sounds appropriate.
4. He runs a configuration dump, a state dump, or a self diagnostic test against the Diagnostic Provider to collect information about the component.
5. From the state dump, the administrator is able to notice that the component state is not what would be expected for its workload.

6. The administrator works with the test team to determine which of the flows is causing the state of the component to diverge from what is expected (as evidenced by repeated execution of the state dump).

For more information on creating Diagnostic Providers, see the WebSphere Application Server, Version 6.1 information center.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 6. Introduction to JVM-related problems

Estimated time

01:00

What this unit is about

This unit provides an overview of common types of JVM-related problems.

What you should be able to do

After completing this unit, you should be able to:

- Describe javacore files and how to obtain them
- Identify a sluggish JVM and detect bottleneck problems
- Explain how to tune the heap size
- Benefit from the WebSphere Education tuning course and other learning resources

How you will check your progress

Accountability:

- Checkpoint

References

SG24-6798-00 *WAS V6 Problem Determination for Distributed Platforms*

JDK Diagnostic Guides

<http://www.ibm.com/developerworks/java/jdk/diagnosis/>

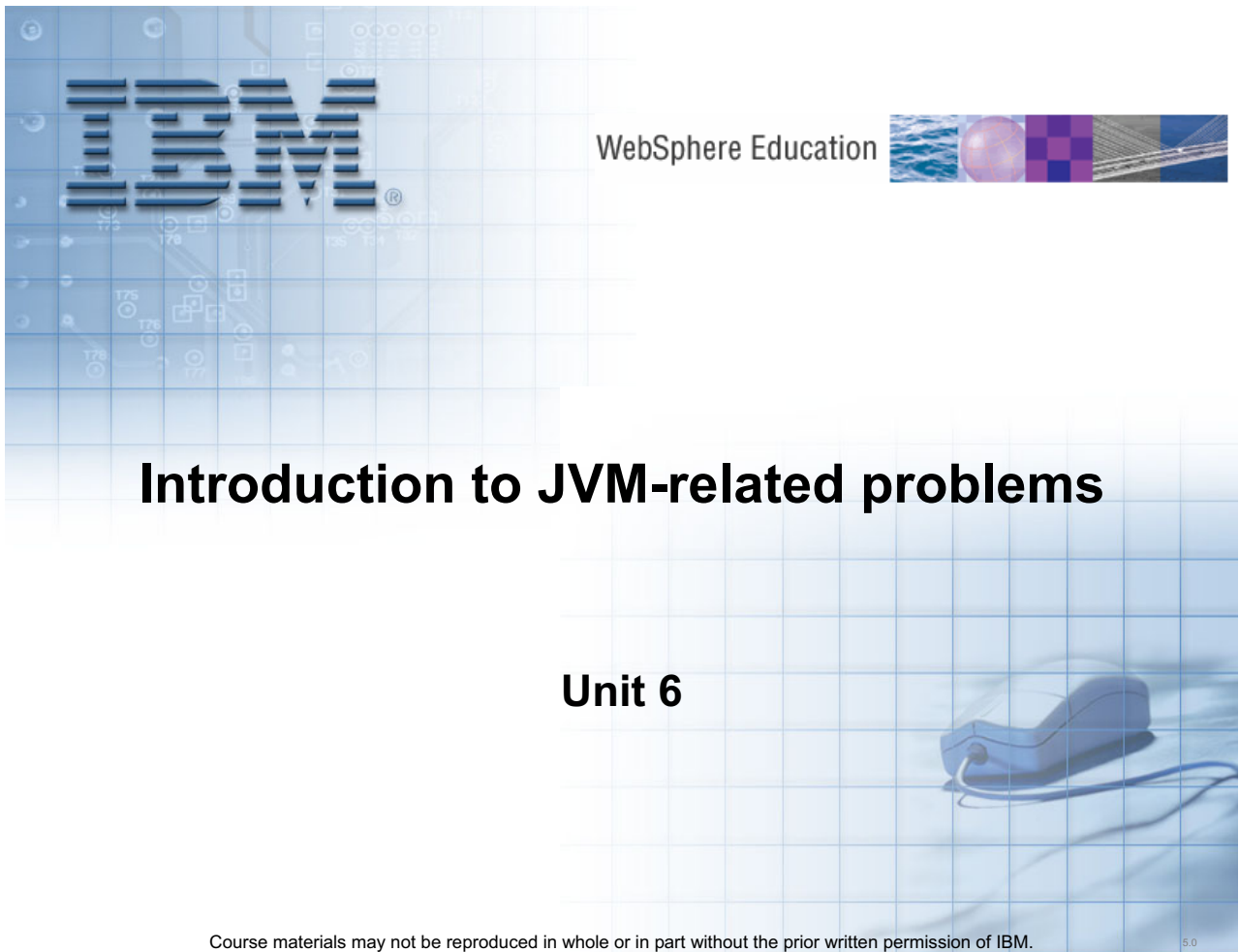


Figure 6-1. Introduction to JVM-related problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — Estimated time

01:00

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe the feature of the JVM and its architecture
- Tune the JVM

Figure 6-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM introduction

After completing this topic, you should be able to:

- Describe the components and functions provided by the JVM
- Describe how the Garbage Collector works

Figure 6-3. JVM introduction

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Java virtual machine (JVM) features

- Almost every WebSphere process runs in a JVM
- The JVM provides:
 - Class loading
 - As the name indicates, a class loader loads and verifies the classes. Multiple class loaders are involved in loading the required libraries for an application to run. Each class must be loaded by a class loader.
 - Garbage collection
 - Garbage collection takes care of memory management for the entire application server. It searches memory to reclaim space from program segments or inactive data.
 - Execution management
 - Manages the bookkeeping work for all the Java threads.
 - Execution engine
 - Interprets the Java methods.

Figure 6-4. Java virtual machine (JVM) features

WA5711.0

Notes:

The Java virtual machine (JVM) is an interpretive computing engine that is responsible for running the bytecode in a compiled Java program. The JVM translates the Java bytecodes into the native instructions of the host machine. The application server, being a Java process, requires a JVM in order to run and to support the Java applications that are running on it. JVM settings are part of an application server configuration.

It is called “virtual” because it provides a machine interface that is independent of the underlying operating system and machine hardware architecture. This independence from hardware and operating system is a cornerstone of the write-once run-anywhere value of Java programs. Java programs are compiled into bytecodes that target the abstract virtual machine; the JVM is responsible for executing the bytecodes on the specific operating system and hardware combinations.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Just-in-time compiler (JIT) basics

- The Just-in-time compiler (JIT) is not really part of the JVM but is essential for a high performing Java application
 - Java is Write Once Run Anywhere thus it is interpreted by nature and without the JIT could not compete with native code applications
- The JIT works by compiling bytecode loaded from the class loader when it is accessed by an application.
 - Due to different platforms having different JITs there is no standard method for when a method is compiled.
 - As your code accesses methods the JIT determines how frequently specific methods are accessed and compiles those touched often quickly to optimize performance

Figure 6-5. Just-in-time compiler (JIT) basics

WA5711.0

Notes:

All the JVMs that are currently used commercially come with a supplementary compiler that takes bytecodes and outputs platform-dependent machine code. This compiler works with the JVM to select parts of the Java program that would benefit from the compilation of bytecode, and replaces the virtualized interpretation of these areas of bytecode with concrete code for the JVM. This is called just-in-time (JIT) compilation.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM version

- WebSphere supports several JVMs based on version and platform type
 - Windows, AIX and Linux – IBM supplied
 - SUN and HP – hybrid of IBM add-ons and vendor supplied JVM
- For a comprehensive list of the supported JVMs check
 - <http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html>
- To determine the JVM version in use:
 - Look in the SystemOut.log file of one of the profile instances
 - `<profile_home>/logs/server1/SystemOut.log`
 - Run **java -fullversion** (IBM JVMs only) or **-version** from the command line
 - `<was_home>/java/bin/java -fullversion`

```
C:\WebSphere\AppServer\java\bin>java -fullversion
java full version "J2RE 1.5.0 IBM Windows 32 build
pwi32dev-20060511 (SR2) "
```

Figure 6-6. JVM version

WA5711.0

Notes:

JVMs differ on the available tools supported and can have different behavior, even between different versions targeted for the same platform. Always look at the documentation for the specific JVM for behavioral descriptions and the options to control the JVM.

WebSphere Application Server V6.1 support the Java Development Kit (JDK) version 5.0. This version can be quite different from JDK version 1.4.2 that is supported by WebSphere Application Server V6.0 and earlier. This new JDK introduces a new Virtual Machine Implementation, a new garbage collection scheme and a new just-in-time (JIT) compiler.

The JVM version matches the JDK level. The JVM is the runtime component of the JDK. The prerequisite page will identify the supported JDKs. The WebSphere-supplied Java 2 SDK is required for both the run time and any remote Java clients for Windows. For AIX and Linux, the supported version of the JDK is IBM 32-bit SDK (or 64-bit SDK where appropriate), Java 2 Technology Edition, v5 SR2. The actual JVM build number is provided which can further identify the exact JVM in use and is helpful in determining the exact JVM being used.

IBM does not supply a software developer kit or runtime environment for HP and Sun platforms. However, IBM does make strategic products, such as the WebSphere Application Server, for these platforms.

The JVM is installed as part of the WebSphere Application Server in the Java directory tree.

For Sun Solaris, WebSphere Application Server contains an embedded copy of the Sun Solaris JVM along with some IBM add-ons, such as security, XML, and ORB packages. The WebSphere Application Server Solaris SDK is, therefore, a hybrid of Sun and IBM products. However, the core JVM and JIT are Sun Solaris.

For HP-UX, WebSphere Application Server contains an embedded copy of the HP JVM alongside some IBM add-ons, such as security packages. The WebSphere Application Server HP SDK is, therefore, a hybrid of HP and IBM products. However, the core JVM and JIT are HP software.

IBM does service these JVMs, but only when it is an embedded part of IBM middleware (for example, WebSphere Application Server).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM overview

- JVM is built using OO design
- The core run times of JVMs are developed in C or C++ and execute a large majority of function in native code
 - Garbage collector/memory management
 - JIT
 - I/O subroutines, OS calls
- The J2SE/J2EE APIs all exist at the Java code layer.
 - Makes data structures available
 - Gives users access to needed function
 - Allow black box interactions with system

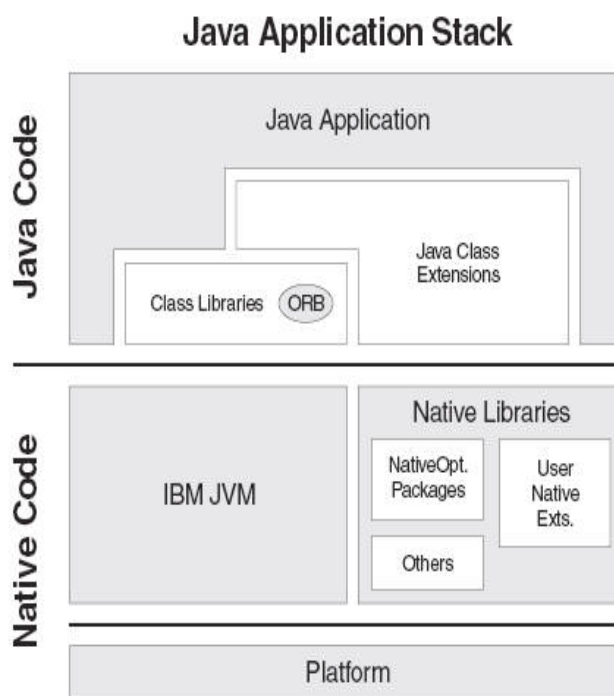


Figure 6-7. JVM overview

WA5711.0

Notes:

Keep in mind that WebSphere Application Server is a Java application.

The just-in-time (JIT) compiler works by compiling bytecode loaded from the class loader when it is accessed by an application

- Due to different platforms having different JITs there is no standard method for when a method is compiled.
- As your code accesses methods the JIT determines how frequently specific methods are accessed and compiles those touched often quickly to optimize performance

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM memory segments

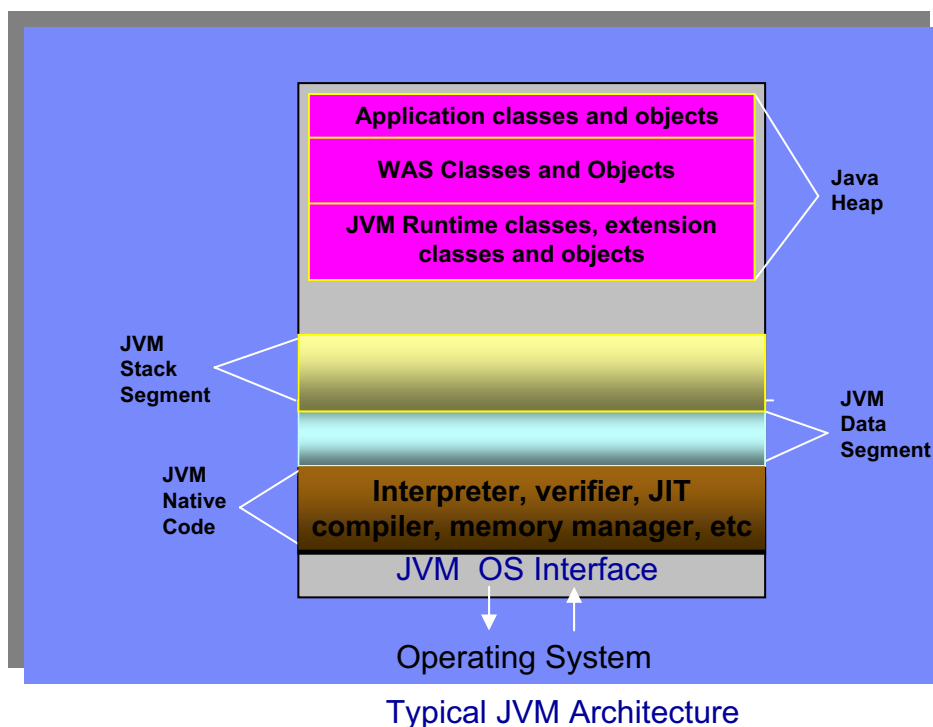


Figure 6-8. JVM memory segments

WA5711.0

Notes:

The important points to remember on this slide:

1. The JVM is a process whose memory layout depends on the OS platform. The diagram above is a typical abstraction of process memory layout.
2. As a process, the JVM itself needs its own heap (JVM native heap) from which it gets its own dynamic memory needs.
3. A portion of that heap is allocated for the Java heap where classes and objects are actually stored. This is the heap that Java programmers are familiar with.
4. Garbage collection occurs only in the Java heap.
5. The Java heap can grow and shrink. When there are enough available memory in the JVM heap, the Java heap can be extended easily without acquiring memory from the OS. Otherwise, a request for additional memory from the OS is made and the JVM process size increases consequently.

The data segment contains a lot of native memory buffers used for the network IO facility (as well as database buffer if using a type 2 JDBC driver).

Memory not allocated to the Java Heap is available to the native heap:

Available process memory space – Java heap = native heap.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Understanding garbage collection (GC)

- JVM manages the Java heap and does garbage collection
- Object is eligible for GC when there are no more references from root to that object
- Root set - References in the stack and registers
- Reference is dropped when variable goes out of scope

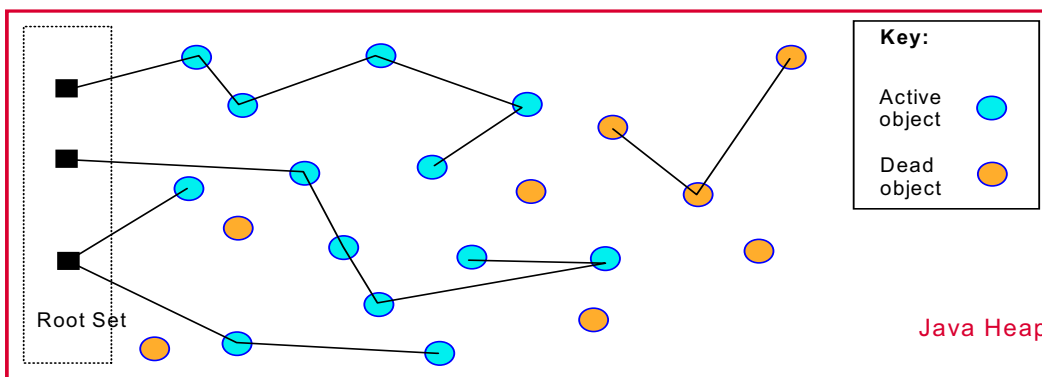


Figure 6-9. Understanding garbage collection (GC)

WA5711.0

Notes:

The active state of the JVM is made up of the set of stacks that represents the threads, the statics that are inside Java classes and set of local and global JNI references.

The GC scans the thread stacks and look for pointers that look like they are pointing to Java objects (that is, within the Java heap range). It is likely that there are attributes (for example, floats) being pushed onto the stack that look like Java object references, that is why GC marks them as “dosed” to make them unmovable during compaction.

The root set also includes pinned objects, JNI references, and so on.

All the objects referenced by these pointers are reachable objects, they will be listed in a mark vector, which will then be used to compare with the allocbits vector. Allocbits vector contains the information the GC needs for all the objects created, the differences between these two vectors the GC can determine what needs to be collected.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM JDK GC process

- **Mark:** Recursively marks all the live objects, starting with the registers and thread stacks.
 - Parallel Mark and Sweep
 - Use multiple threads (no. processors -1) to perform tasks
- **Sweep:** Frees all the objects that were not marked in the mark phase.
- **Compaction:** Reduces heap fragmentation. This phase attempts to move all live objects to one end of the heap, freeing up large areas of contiguous free space at the other end.
 - Compaction stops JVM activity while it occurs
 - Not every GC cycle results in a compaction

Figure 6-10. IBM JDK GC process

WA5711.0

Notes:

The IBM Garbage Collector is by default a “stop-the-world” (STW) operation, because all application threads are stopped while the garbage is collected.

Concurrent mark can be configured. It starts a concurrent marking phase before the heap is full. The mark phase runs while the application is still running, in effect, attempting to trade application performance for possible smaller garbage collection times.

Mark stack overflow (MSO) is a rare event which can occur. Because the mark stack has a fixed size, it can overflow. This has a negative impact on pause time:

- Mark process resumes where the overflow occurred
- Repeats <n> times until no more objects requiring marks

A parallel version of Garbage Collector Mark has been developed. The time spent marking objects is decreased through the addition of helper threads and a facility that shares work between those threads.

Incremental compaction is a way of spreading compaction work across a number of garbage collection cycles, thereby reducing pause times. Another important task for

Incremental Compaction is the removal of dark matter. Dark matter is the term for small pieces of free space (currently less than 512 bytes in size) that are not on the free list and therefore are not available for allocation of objects.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Garbage collection policies

Memory management is configurable using four different policies with varying characteristics.

- Optimize for Throughput – flat heap collector focused on maximum throughput.
- Optimize for Pause Time – flat heap collector with concurrent mark and sweep to minimize GC pause time.
- Generational Concurrent – divides heap into “nursery” and “tenured” segments providing fast collection for short lived objects. Can provide maximum throughput with minimal pause times.
- Subpool – a flat heap technique to help increase performance on large SMP systems with 16 or more processors by optimizing the object allocation. Only available on IBM pSeries and zSeries

Figure 6-11. Garbage collection policies

WA5711.0

Notes:

The default policy is optimize for throughput. It is a “stop-the-world” policy where no other work is performed by the JVM. To optimize the throughput, the garbage collection performs all of its work during the GC cycle and does not effect the application when not active.

The `-Xgcpolicy` command line parameter is used to enable and disable concurrent mark:

```
-Xgcpolicy:<optthruput | optavgpause | gencon | subpool>
```

The `-Xgcpolicy` options have these effects:

- ***optthruput*** Disables concurrent mark. If you do not have pause time problems (as seen by erratic application response times), you get the best throughput with this option. *Optthruput* is the default setting.
- ***optavgpause*** Enables concurrent mark with its default values. If you are having problems with erratic application response times that are caused by normal garbage collections, you can reduce those problems at the cost of some throughput, by using the *optavgpause* option.

- **gencon** Requests the combined use of concurrent and generational GC to help minimize the time that is spent in any garbage collection pause.
- **subpool** Disables concurrent mark. It uses an improved object allocation algorithm to achieve better performance when allocating objects on the heap. This option might improve performance on large SMP systems. The *subpool* option is available only on AIX, Linux PPC and zSeries, z/OS, and i5/OS.

Instructor notes:

Purpose —

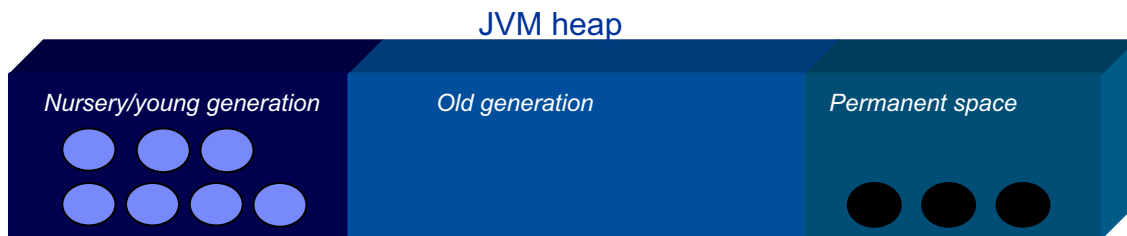
Details —

Additional information —

Transition statement —

Generational garbage collection

- How the IBM generational and Sun/HP Garbage Collectors work
- Not the default garbage collection policy



- Minor collection – Takes place only in the young generation, normally done through direct copying (very efficient)
- Major collection – Takes place in the old generation and uses the normal *mark and sweep* algorithm

Figure 6-12. Generational garbage collection

WA5711.0

Notes:

The Generational Concurrent Garbage Collector has been introduced in Java 5.0 from IBM. A generational garbage collection strategy is well suited to an application that creates many short-lived objects, as is typical of many transactional applications.

Different algorithms used in each generation. For young generations compaction or by copy algorithms are used. For older tenured generations the normal mark and sweep algorithms are used.

The Java heap is split into two areas, a new (or nursery) area and an old (or tenured) area. Objects are created in the new area and, if they live long enough, they are moved or promoted into the old area. Objects are promoted when they reach a certain age (known as the tenure age). This age is a count of the number of garbage collections for which they have lived.

Tenure age is a measure of the object age at which it should be promoted to the tenure area. This age is dynamically adjusted by the JVM and reaches a maximum value of 14. The age of an object is incremented on each scavenge. A tenure age of x implies that, if the object has survived x flips between survivor and allocate space, it is promoted. The

threshold is adaptive and adjusts the tenure age based on the percentage of space used in the new area.

Instructor notes:

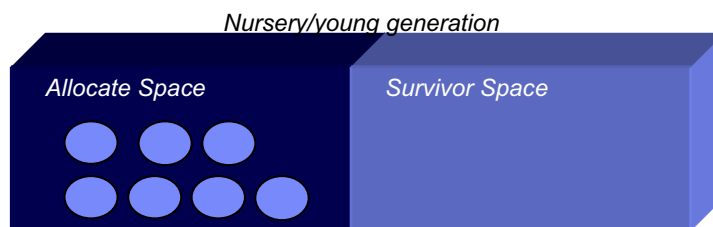
Purpose —

Details —

Additional information —

Transition statement —

Nursery/young generation



- Nursery is split into two spaces (semi-spaces)
 - Only one contains live objects and is available for allocation at a time
 - Minor collections (scavenges) move objects between spaces
 - Role of spaces is reversed
- Movement results in implicit compaction, reducing fragmentation

Figure 6-13. Nursery/young generation

WA5711.0

Notes:

The new area is split into two logical spaces: allocate and survivor. Objects are allocated into the allocate space. When that space is filled, a GC process called scavenge is triggered. During a scavenge, live objects are copied either into the survivor space or into the tenured space if they are old enough to have reached the tenured age. Then the roles are reversed, the survivor space becomes the allocate space.

Dead objects in the nursery remain untouched. When all the live objects have been copied, the spaces in the new area switch roles. The new survivor space is now entirely empty of live objects and is available for the next scavenge. Any objects that were not moved into the old/tenured space are now in the “new” allocate space.

The size of the allocate space in the new area is maximized by a technique called *tilting*. Tilting controls the relative sizes of the allocate and survivor spaces. Based on the amount of data that survives, the ratio can be adjusted to make the survivor space smaller.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM problem determination introduction

After completing this topic, you should be able to:

- Articulate at a high level, the common JVM problems and how to begin problem determination
- Describe a javacore
- Collect and read javacore files

Figure 6-14. JVM problem determination introduction

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM problem determination: Symptom analysis

- Application server stops responding
 - Server crash
 - Application server has terminated
 - Tools available to determine the cause of crash
 - Unexpected process exit
 - Hung process
 - Verify application server process is still running
 - Tools available to determine which process is hung
 - Deadlocks
 - Looping code logic
 - Out of memory condition
 - Errors and exceptions logged without process exit
 - At time may result in unexpected process exit
- Performance degradation
 - Check to if the process ID is continually changing
 - Indicates the application server is probably crashing and being restarted

Figure 6-15. JVM problem determination: Symptom analysis

WA5711.0

Notes:

An application server that does not respond might be hung or the process might have ended. Users see hung applications or are not able to access new applications.

An application server can suffer performance degradation when an application server is repeatedly crashing and being restarted automatically.

If CPU activity is low but the application server has not terminated, you most likely have a hang or deadlock situation. If CPU activity is high and the application server is using the cycles, you most likely have a loop or inefficient code.

An often encountered condition is out-of-memory (OOM), which manifests itself either as an unexpected process exit with a "OutOfMemoryException" or just a bunch of errors and exceptions but no immediate process exit. This can occur when the application server is running low on memory, perhaps due to an application issue such as a memory leak, a hardware memory failure, or higher than usual demand.

Server crashes, hung processes and out of memory will be covered in detail in subsequent units.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM problem determination: Data to collect

- Core files
 - Also known as process dumps or system core files
 - Complete dump of the virtual memory
 - Can be quite large
 - May be required by IBM support
 - Tools available to parse these files

- Javacore files
 - Also known as javadump or thread dump files
 - Text file created by an application server during a failure
 - Can also be generated manually
 - Error condition will be given at top of dump
 - Specific to the IBM JDK

- VerboseGC logs
 - Provides detailed information on garbage collection cycles

- Heap dump
 - Shows the objects using the heap memory
 - Needed for memory leak determination

Figure 6-16. JVM problem determination: Data to collect

WA5711.0

Notes:

These data artifacts will be covered in more detail in later units. They are presented here as an overview on the types of data available.

Javacore creation is enabled by default, can be disabled with option “DISABLE_JAVADUMP”. The steps to initiate and control the output of the javacore file vary by JVM type. It always best to look at the documentation provided with the specific JVM. On Sun and HP platforms the thread dump goes to native_stdout.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore overview

- What is a javacore?
 - Small diagnostic file that is produced by the JVM
 - A text file that contains a lot of vital information about the running JVM process
 - javacore lists the JVM command line, environments, loaded libraries, etc.
 - Provides a snapshot of all the running threads, their stack traces and the monitors (locks) held by the threads.
 - GC history and storage management (memory) information
 - Can be a key in detecting hang/deadlock conditions.

- Also can be helpful in detecting performance problems
 - Take a few (at least three) snapshots of the JVM (about 2-3 minutes apart)
 - Analyze the javacore to see what different threads are doing in each snapshot
 - Example: A series of snapshots where container threads are in the same method or waiting on same monitor or resource would be an indication of a bottleneck, hang or a deadlock.

Figure 6-17. Javacore overview

WA5711.0

Notes:

A javacore file is a snapshot of a running Java process. The IBM Java SDK produces a javacore in response to specific operating system signals. Javacore files show the state of every thread in the Java process, as well as the monitor information identifying Java synchronization locks. If deadlocks are detected this will be noted in the file.

It provides various storage management values (in hexadecimal), including the free space in heap, the size of current heap, and details on other internal memory that the JVM is using. It also contains garbage collection history data. This is a new feature in JDK 5.

You can use javacore files to resolve the problems listed below. For more information on capturing data for problem determination, see the MustGather documents for these problems:

- 100% CPU usage
- Crash
- Hang or performance degradation

Javacore files can also provide an initial insight for memory issues.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore file location and naming

- The javacore file is stored in the first viable location of:
 - The setting of the IBM_JAVACOREDIRE environment variable
 - `<WAS_install_root>/profiles/<profile>`
 - TMPDIR or TEMP environment variable
 - Windows only: If the javacore cannot be stored in any of the above, it is put to STDERR.

- Javacore naming
 - Windows and Linux: javacore.YYYYMMDD.HHMMSS.PID.txt where YYYY=year, MM=month, DD=day, SS=second, PID=processID
 - AIX: javacorePID.TIME.txt where PID=processID, TIME=time since 1/1/1970
 - z/OS: Javadump.YYYYMMDD.HHMMSS.PID.txt where YYYY=year, MM=month, DD=day, SS=second, PID=processID

Figure 6-18. Javacore file location and naming

WA5711.0

Notes:

A javacore file can be generated on demand through the wsadmin command line interface:

1. From the command prompt, enter the command **wsadmin.bat** to get a wsadmin command prompt.



Note

If security is enabled or the default SOAP ports have been changed, you will need to pass additional parameters to the batch file in order to get a wsadmin prompt. For example:

```
wsadmin.bat [-host host_name] [-port port_number] [-user userid[-password password]]
```

**Note**

You can connect wsadmin to any of the server JVMs in the cell. After running the wsadmin command it will display the server process for which it has attached to. Depending on the process that it has attached to, you can get thread dumps for various JVMs. If wsadmin is connected to deployment manager, then you can get thread dumps for any JVM in that cell. If it is attached to a node agent, then you can get thread dumps for any JVM in that Node. If it is attached to a server, then you can get thread dumps only for the server to which has connected to.

2. Get a handle to the problem application server (Note that the contents in brackets "[.....]", along with the brackets, is not optional. It must be entered to set the JVM object. Also, note that there is a space between *completeObjectName* and *type*.):

```
wsadmin> set jvm [ $AdminControl completeObjectName  
type=JVM,process=server1,* ]
```

Where *server1* is the name of the application server that does not respond (or is *hung*). If wsadmin is connected to a dmgr and if the server names in cell is not unique, the you can qualify the JVM with node attribute in addition to process.

3. Generate the thread dump:

```
wsadmin> $AdminControl invoke $jvm dumpThreads
```

The IBM_JAVACOREDIR depends on the JVM version and platform. For detailed information check the IBM JDK Diagnostic Guide, which explains in detail all the options for controlling javacores such as naming and location.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore file subcomponents

- **TITLE:** Shows basic information about the event that caused the generation of the Javadump, the time it was taken, and the file name.
- **GPINFO:** Shows some general information about the operating system. If the dump was caused by a general protection fault (gpf), information about the failure is provided. Namely the fault module is identified.
- **ENVINFO:** Shows information about the JRE level, details about the command line that launched the JVM process and the JVM environment.
- **MEMINFO:** Shows the free space in heap, the size of current heap, and details on other internal memory that the JVM is using. It also contains garbage collection history data.
- **LOCKS:** Shows the locks threads hold on resources including other threads. A lock (also referred to as a monitor) prevents more than one entity from accessing a shared resource.
- **THREADS:** Identifies the current thread, provides a complete list of Java threads that are alive and provides their stack traces.
- **CLASSES:** Shows information on the classloaders used and specific classes loaded

Figure 6-19. Javacore file subcomponents

WA5711.0

Notes:

GPF stands for general protection fault.

Use the Java command line to determine if the javacore is for a WebSphere Application Server Java process. If so, which WebSphere Java process it is for: an Application Server process, an Administrative Server process, a jmsserver, a node agent, or some other WebSphere Application Server process.

The current thread is the thread that is running when the signal is raised that causes the javacore to be written. The **Current Thread Details** shows the Java and native stacks of the current thread. Since these are stacks, the most recent calls are at the top of the stack.

- The Java stack shows the Java method calls that are made by the current thread.
- If there is a native stack, this contains the most recent events that the thread executed. As the name implies, these are events in native code (libraries) called by Java

The javacore processing dumps the current stack for every thread in the JVM. It shows the current state of the thread, that is whether it is *Runnable* or not. It also shows the thread

name, which is useful in determining how that thread is used. It also shows the Java stack and native stack for each thread.

It is important to understand what signal caused the javacore. This determines how to interpret the information in the javacore file. Does the signal indicate a JVM crash? Does the signal information state it was due to an operator action (noted as “user” dump event)? If the javacore is written to diagnose a hung JVM process, the steps you take are very different than if the javacore is due to a JVM crash.

Each line of information will be identified by tags, such as **TISIGINFO**. Different sections contain different tags, which make the file easier to parse for performing simple analysis.

Normal tags have these characteristics:

- Tags are up to 15 characters long (padded with spaces).
- The first digit is a nesting level (0,1,2,3).
- The second and third characters identify the component that wrote the message. The major components are:
 - CI command line interpreter
 - CL Class loader
 - LK Locking
 - ST Storage (Memory management)
 - TI Title
 - XE Execution engine
- The remainder is a unique string.
- Special tags have these characteristics:
 - A tag of NULL means the line is just to aid readability.
 - Every section is headed by a tag of 0SECTION with the section title.

To verify that the javacore file is complete you should see a `END OF DUMP` string at the end of the file.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore example (JDK v5)

```

NULL -----
0SECTION      TITLE subcomponent dump routine
NULL =====
1TISIGINFO    Dump Event "user" (00004000) received
1TIDATETIME   Date:                2007/08/24 at 02:00:55
1TIFILENAME    Javacore filename:
c:\WebSphere\AppServer\profiles\profile01\javacore.20070824.020055.67192.txt
NULL -----
0SECTION      GPINFO subcomponent dump routine
NULL =====
2XHOSLEVEL    OS Level                : Windows 2000 5.0 build 2195 Service Pack 4
2XHCPUS       Processors -
3XHCPUARCH    Architecture      : x86
3XHNUMCPUS    How Many         : 1
NULL
1XHERROR2     Register dump section only produced for SIGSEGV, SIGILL or SIGFPE.
NULL
NULL -----
0SECTION      ENVINFO subcomponent dump routine
NULL =====
1CIJAVAVERSION J2RE 5.0 IBM J9 2.3 Windows 2000 x86-32 build j9vmwi3223-20060504
1CIVMVERSION  VM build 20060501_06428_lHdSMR
1CIJITVERSION JIT enabled - 20060428_1800_r8
1CIRUNNINGAS  Running as a standalone JVM
1CICMDLINE    c:/WebSphere/AppServer/java/bin/java -Declipse.security - {deleted text}
1CIJAVAHOMEDIR Java Home Dir:   c:\WebSphere\AppServer\java\jre
1CIJAVADLLDIR Java DLL Dir:    c:\WebSphere\AppServer\java\jre\bin
1CISYSCP      Sys Classpath:  c:/1CIUSERARGS   UserArgs: {deleted text}
2CIUSERARG    -Xjcl:jclscar_23
2CIUSERARG    -
2CIUSERARG    -Dsun.boot.library.path=c:\WebSphere\AppServer\java\jre\bin
{rest of file deleted}

```

Figure 6-20. Javacore example (JDK v5)

WA5711.0

Notes:

For example, the above indicates that the javacore file was created as the result of a “user” dump event which indicates that this file was created manually and not due to an error.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore example (JDK 1.4.2 and earlier)

```

NULL -----
0SECTION      TITLE subcomponent dump routine
NULL          =====
1TISIGINFO    signal 11 received
1TIDATETIME   Date:                2004/04/06 at 16:04:16
1TIFILENAME   Javacore
filename:     /export/home/was4/wasapps/appvmc01/vmclaimt1/workdir
             /javacore499836.1081281856.txt
NULL          -----
0SECTION      XHPI subcomponent dump routine
NULL          =====
1XHTIME       Tue Apr  6 16:04:16 2004
1XHSIGRECV    SIGSEGV received at 0xd2782a88 in
             /opt/WebSphere4/AppServer/java/jre/bin/libjitc.a. Processing
             terminated.
1XHFULLVERSION J2RE 1.3.1 IBM AIX build ca131-20030630
.....

```

Figure 6-21. Javacore example (JDK 1.4.2 and earlier)

WA5711.0

Notes:

For example, the following message indicates a crash in a JVM library and should be pursued with WebSphere Application Server Support.

Crash in the libjitc.a library 1XHSIGRECV SIGSEGV received at 0xd2782a88 in /opt/WebSphere4/AppServer/java/jre/bin/libjitc.a. Processing terminated.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Verbose GC

- Verbose GC is an option provided by the JVM run time
- Enables a garbage collection log
 - Interval between collections
 - Duration of collection
 - Compaction required
 - Memory size/memory freed/memory available
- Enable the verbose GC for the server using the administration console
 - Navigate to **Servers -> Application servers -> <serverName>**
 - Under Server Infrastructure, click **Java and Process Management -> Process Definition -> Java Virtual Machine**
 - Select **Verbose Garbage Collection** check box
 - Save and distribute
 - Restart the servers
- Usually writes to native_stderr
 - Varies depending on platform and WebSphere version
 - Some overhead because of disk I/O, but usually minimal unless thrashing

Figure 6-22. Verbose GC

WA5711.0

Notes:

It is often recommended to have verbose GC enabled permanently in production. The overhead cost on a reasonably well-tuned JVM is actually quite small. The benefits of having it on the first time something happens are considerable (no need to reproduce the problem a second time after enabling). It is also good to keep an eye on the verbose GC regularly simply as a way to monitor the health of the system, even when nothing bad has been noticed.

So of course enabling verbose GC by default is a decision that must be made consciously by each system administrator. But it is no longer plainly "not recommended as a normal production setting."

In version 6.1, it is possible to enable verbose GC output using the **Runtime** tab. This allows the initiation of verbose GC output on a running application server.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM tuning

After completing this topic, you should be able to:

- Present a tuning methodology
- Understand tuning considerations

Figure 6-23. JVM tuning

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tuning methodology

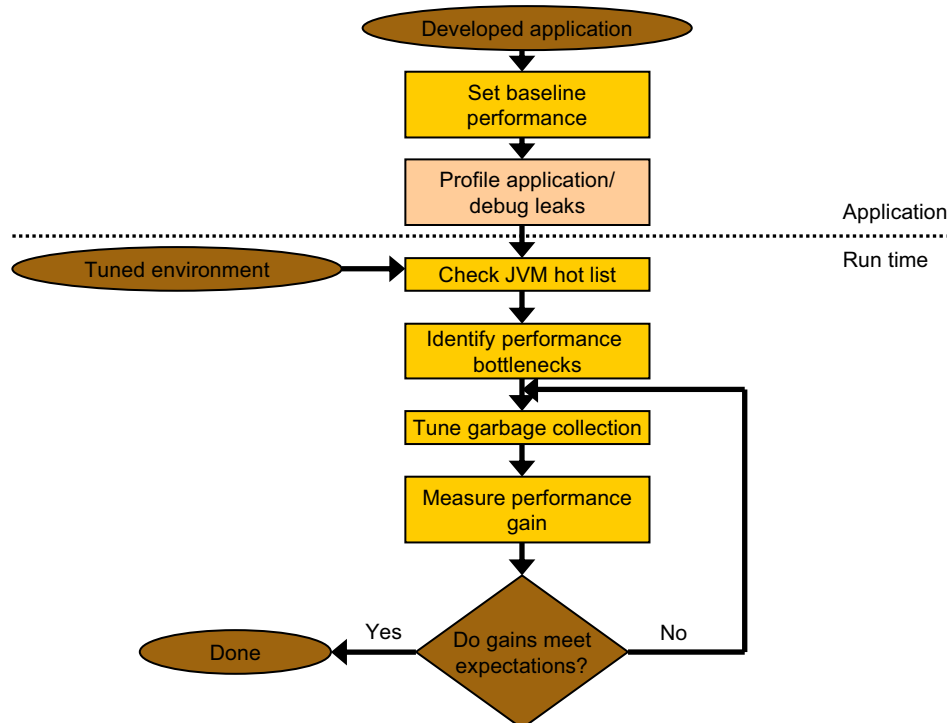


Figure 6-24. Tuning methodology

WA5711.0

Notes:

The tuning process is iterative in nature. To find the optimal configuration several tests and evaluation should be performed.

There are a lot of tuning parameters available to control the behavior of the JVM. These are discussed in detail in the JDK Diagnostic Guide. It is worth noting that the majority of JVM tuning involves configuring the ideal heap size and to a lesser degree the gc policy. It is encouraged to concentrate on these two areas. The other parameters provide fine tuning and are beyond the scope of this class.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tuning considerations: Heap

- Start with a reasonable maximum heap (-Xmx) size
 - 512M for WebSphere Application Server
- Test different maximum heap sizes to find optimal setting
 - Opposing forces
 - The larger the heap, typically the longer the GC cycle
 - The smaller the heap, the more frequently GC is required
 - Size maximum heap somewhat larger than steady state to allow for peak load
- Setting minimum heap size (-Xms) can improve server startup time
 - If using default, may need to be resized several times during startup
- Setting the minimum too large may impact runtime performance
 - Larger value means more memory space requiring garbage collection
 - The JVM cannot compensate if you make a poor choice

Figure 6-25. Tuning considerations: Heap

WA5711.0

Notes:

It is worth noting that with a min heap size different from the max heap size, the JVM is allowed to resize the heap over time to adjust to changing conditions. Even among the experts, opinions vary on whether that is a good thing or a bad thing.

Proper JVM heap sizing is usually the only major area of WebSphere Application Server requiring tuning:

- Too little heap
 - Causes the application to GC frequently
 - Consumes CPU
 - Extreme cases can lead to error conditions and server failure
- Too much heap
 - Wasteful
 - May cause long response times, especially on Sun

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tuning considerations: Garbage collection

- “Throughput” (the JVM definition)
 - Measure of productive time excluding time spent in GC.
- Pauses
 - Measure of time when application execution pauses during GC
- Turn on `verbosegc` for more information on GC operation
- Avoid calling **`System.gc()`** from the application
 - Causes the least efficient, slowest GC to take place
 - Always triggers a compaction
 - **`System.gc()`** does not always trigger an immediate GC
 - Allow the JVM to determine optimum time to GC
 - Can be disabled in IBM JDKs using `-Xdisableexplicitgc`

Figure 6-26. Tuning considerations: Garbage collection

WA5711.0

Notes:

When analyzing GC cost, two metrics must be investigated: frequency and duration. GC is the main cause of memory-related performance bottlenecks in Java.

There are two primary measures of garbage collection performance. *Throughput* is the percentage of total time not spent in garbage collection, considered over long periods of time. Throughput includes time spent in allocation (but tuning for speed of allocation is generally not needed.) *Pauses* are the times when an application appears unresponsive because garbage collection is occurring.

Users have different requirements of garbage collection. For example, some consider the right metric for a Web server to be throughput, since pauses during garbage collection may be tolerable, or simply obscured by network latencies. However, in an interactive graphics program even short pauses may negatively affect the user experience.

Some users are sensitive to other considerations. *Footprint* is the working set of a process, measured in pages and cache lines. On systems with limited physical memory or many processes, footprint may dictate scalability. *Promptness* is the time between when an

object becomes dead and when the memory becomes available, an important consideration for distributed systems, including remote method invocation (RMI).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tuning considerations: GC policy

- “I want my application to run to completion as quickly as possible.”
 - `-Xgcpolicy:optthruput`
- “My application requires good response time to unpredictable events.”
 - `-Xgcpolicy:optavgpause`
- “My application has a high allocation and death rate (objects are short-lived).”
 - `-Xgcpolicy:gencon`
- “My application is running on big metal and has high allocation rates on many threads.”
 - `-Xgcpolicy:subpool`

Figure 6-27. Tuning considerations: GC policy

WA5711.0

Notes:

The *subpool* option (available on AIX, Linux PPC and zSeries, i5/OS, and z/OS platforms only) uses an improved object allocation algorithm to achieve better performance when allocating objects on the heap. This option might improve performance on large SMP systems.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tuning considerations: Native heap

- Beware: The GC does not look into the native heap
- Ensure JVM is not being swapped out of memory to disk
 - Total Memory Used by JVM > Maximum Heap Size
 - Native memory allocated in addition to the Java heap
 - Native objects include:
 - Database connections for Type 2 JDBC Drivers
 - Thread stacks
 - Compiled methods
 - JNI code and more
 - Physical Memory Available > Total Memory Used by JVM
 - JVMs do not page effectively due to garbage collection

Figure 6-28. Tuning considerations: Native heap

WA5711.0

Notes:

Java Database Connectivity (JDBC) driver types:

- Type 2: Uses native C++ drivers and bridge to JDBC
- Type 4: Uses 100% pure Java drivers that communicate directly to the database using the native protocol of the database.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM tool overview

- WebSphere internal thread pools and heap usage statistics
 - Tivoli Performance Viewer (TPV)

- Thread activity snapshot - Use javacore files
 - Thread Analyzer
 - Thread and Monitor Dump Analyzer

- Memory and garbage collection – Use verbose GC output
 - Extensible Verbose Toolkit (EVTk)
 - IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT)

- Memory utilization by object – Use heap dumps
 - Memory Dump Diagnostic Tool for Java (MDD4J)
 - IBM HeapAnalyzer

Figure 6-29. JVM tool overview

WA5711.0

Notes:

Most tuning and debugging can be accomplished with free tools that many customers have on-hand or ones available using ISA. In addition, high-end tools can also be used such as IBM Tivoli Composite Application Manager for WebSphere or other 3rd party products.

There are tools available to parse system core files.

Most of these tools are covered in more detail in subsequent units.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Some useful Web addresses and resources

- IBM JVM Diagnosis documentation:
 - <http://www.ibm.com/developerworks/java/jdk/diagnosis/>
- IBM JRE and JDK forum:
 - http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=367&start=0&thRange=15&cat=10
- Memory leak detection and analysis in WebSphere Application Server
 - http://www.ibm.com/developerworks/websphere/library/techarticles/0606_poddar/0606_poddar.html
- Garbage collection policy and tuning article on developerWorks
 - <http://www.ibm.com/developerworks/java/library/j-ibmjava3/>
- IBM Guided Activity Assistant (IGAA)
 - IGAA contains a lot of content for JVM problem determination

Figure 6-30. Some useful Web addresses and resources

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Related courses and redbooks

- WF881 course
 - *IBM WebSphere Application Server V6 Performance Monitoring and Tuning for Administrators*

- IBM Redbooks
 - *WebSphere Application Server V6 Problem Determination for Distributed Platforms, SG24-6798-00*

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe the features of the JVM and its architecture
- Tune the JVM

Figure 6-32. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 7. How to troubleshoot hangs

Estimated time

00:30

What this unit is about

This unit looks at how to detect and troubleshoot a hang condition.

What you should be able to do

After completing this unit, you should be able to:

- Describes what a hang is
- Detect a hang condition
- Analyze javacore files for hangs
- Use the WebSphere Application Server hang detection facility
- Use the Thread Analyzer

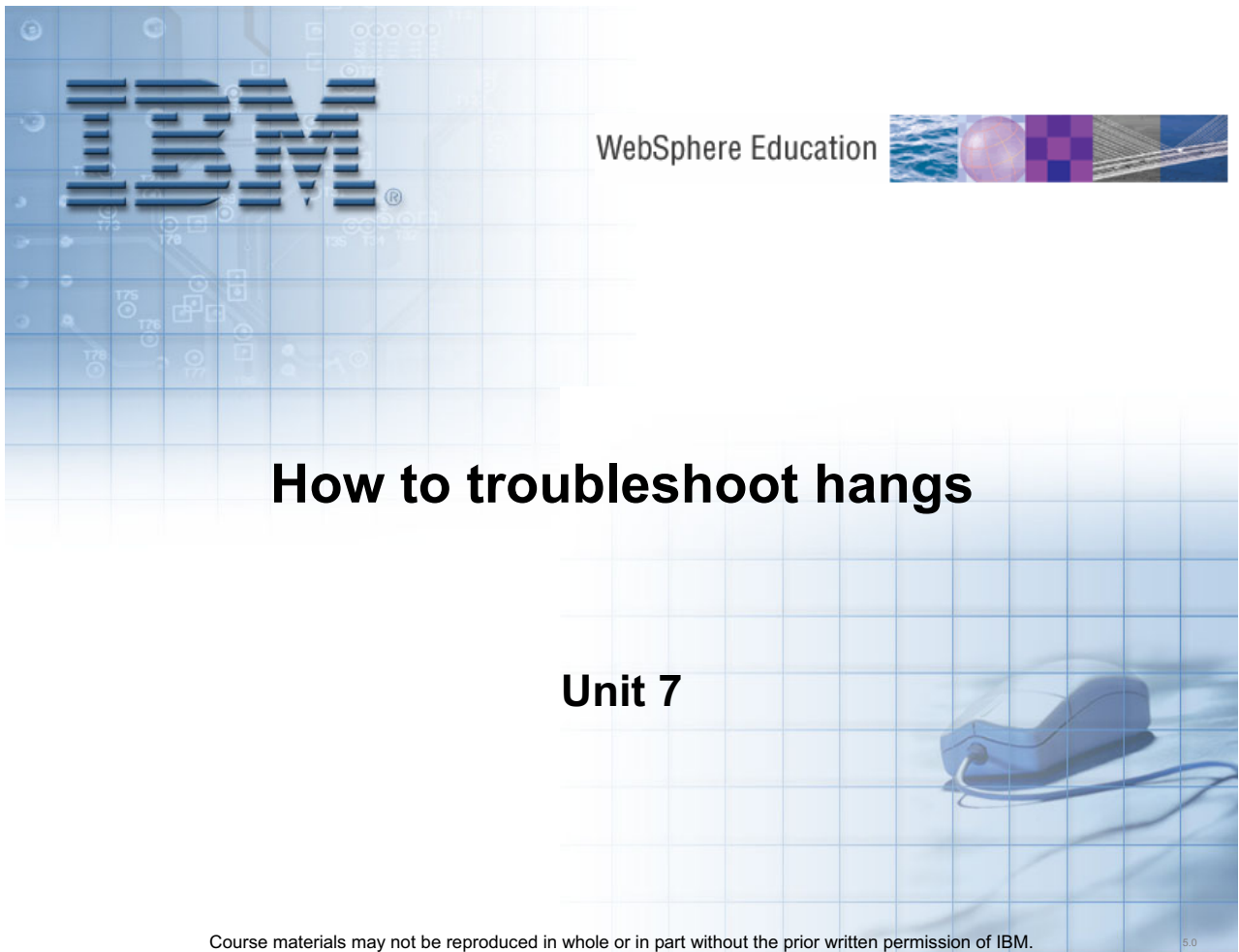
How you will check your progress

Accountability:

- Machine exercises

References

SG24-6798-00 WAS V6 Problem Determination for Distributed Platforms



How to troubleshoot hangs

Unit 7

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.0

Figure 7-1. How to troubleshoot hangs

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe what a hang is and be able to detect one
- Analyze javacore files for hangs
- Use the WebSphere Application Server hang detection facility
- Use the ThreadAnalyzer

Figure 7-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Application server hangs

After completing this topic, you should be able to:

- Describe a hang thread condition
- Articulate how to detect hung threads
- Analyze a javacore file to locate hung threads

Figure 7-3. Application server hangs

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Application server hang defined

- Clarify the nature and extent of the hang
 - A “hang” is not the same as a “crash”
 - Is entire process is truly hung, or does it still respond to some requests?
 - Test with sample “Snoop” servlet, wsadmin commands, and so on.
 - Deadlocks:
 - Very often, one process fails to respond to a request because it has made a call to another process that is itself hung; sometimes is hard to find the true culprit.
 - Deadlocks also may occur within the same Java process, where one thread is deadlocked on another.

Figure 7-4. Application server hang defined

WA5711.0

Notes:

System resources, such as CPU time, might be consumed by this hung transaction when threads run unbounded code paths, such as when the code is running in an infinite loop. Alternately, a system can become unresponsive even though all resources are idle, as in a deadlock scenario. Unless a user or a monitoring tool reports the problem, the system may remain in this degraded state indefinitely.

It is worth noting that some deadlocks or other simpler hangs are contained entirely within one process where all the threads and monitors involved are in the same JVM process and should be visible in a single javacore. Other deadlocks may involve multiple processes. For example, one thread in process A waits for some response from process B, and some thread in process B waits for a response from process A.

In the latter case, it may be difficult to see the full extent of the deadlock or blockage by looking at just one JVM. When you look at a javacore from that type of situation, you may just see a bunch of threads that have made some remote request and are waiting for responses. You may need to look at the other remote processes to fully understand what is

going on in that case. One typical example is where many threads in the server are all waiting for some response from the database.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread hangs

- WebSphere Application Server threads may become hung due to running poorly coded J2EE applications.
- WebSphere administrators may not know the application is slow or hung until a customer calls and complains.
- WebSphere administrators should be alerted before significant problems arise.
- Detecting if a thread is hung or just taking a long time to respond can be a difficult problem to solve correctly.

Figure 7-5. Thread hangs

WA5711.0

Notes:

There are tools available and WebSphere has a built-in monitor to assist in identifying hung threads. These tools and the monitor will be discussed further in the unit. Using these facilities, it can be a simple process to identify when threads are hung.

Keep in mind, not all hangs are the result of application coding problems. They may be the result of problems in the JVM with WebSphere.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread hang examples

- Understand root causes
 - Customer code enters an infinite loop such as:

```
while (true) {  
    i++;  
}
```

Customer code waits infinitely such as:

```
run() {  
    object1.wait();  
}
```

- Customer code creates a deadlock such as:

```
synchronized (object1) {  
    synchronized (object2) {  
        // Work with objects  
    }  
}  
  
synchronized (object2) {  
    synchronized (object1) {  
        // Work with objects  
    }  
}
```

Figure 7-6. Thread hang examples

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere process hang detection steps

- Once a hang is suspected, obtain a thread dump or javacore
 - If the process is still responsive to JMX commands (service threads), then `wsadmin` to trigger the dump
 - Otherwise, may need to trigger through lower-level OS functions
 - On UNIX, send a `kill -3` signal
 - If that fails, may need even lower-level functions (such as `dbxtrace`) or trigger a system core dump for analysis.
- For a typical hang, collect three dumps at a few minutes interval
 - To see if anything is moving within the process (but slowly)
- Examine the thread dumps with Thread Analyzer or by hand
 - Look for deadlocks
 - Look for threads that are waiting after sending a request to some other process, now awaiting a response

Figure 7-7. WebSphere process hang detection steps

WA5711.0

Notes:

Javadumps are enabled by default. Javacore production can be turned off using `-Xdump:java:none`. This is not recommended as Javadumps are an essential diagnostics tool. Use the `-Xdump:java` option to give more fine-grained control over the production of Javadumps.



Note

The use of `-Xdump` is new to JDK 5. For earlier versions of the IBM JDK the javacore can be disabled by setting the `DISABLE_JAVADUMP` environment variable to `TRUE`.

By default, a javacore is triggered when one of the following occurs:

- A fatal native exception occurs in the JVM (not a Java exception).
- The JVM has insufficient memory to continue operation. Often caused by heap expansion and compaction.

- You send a signal to the JVM from the operating system.
- You use the **JavaDump()** method within Java code that is being executed.

The exact conditions in which you get a javadump vary depending on whether the default dump agents have been overridden. A fatal exception is one that causes the JVM to terminate. The JVM handles this by producing a javacore file and then terminating the process. In the user-controlled cases (the latter two), the JVM pauses, performs the dump, and then continues execution.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

How to manually trigger a thread dump

- Use operating system facilities:
 - `kill -3 <WAS PID>` (UNIX/Linux)
- Explicitly tell WebSphere to generate a thread dump
 - From a command prompt start the wsadmin shell by typing: `wsadmin`
 - Execute the following wsadmin commands:

```
wsadmin> set jvm [$AdminControl completeObjectName
  type=JVM,process=<SERVER_NAME>,*]
wsadmin>$AdminControl invoke $jvm dumpThreads
```

Where `<SERVER_NAME>` is the name of the application server

Figure 7-8. How to manually trigger a thread dump

WA5711.0

Notes:

The default location for the placement of the javacore file is:

`<WAS_install_root>/profiles/<profile>`. See the JVM Introduction unit for complete details.

A javacore file can be generated on demand through the wsadmin command line interface:

1. From the command prompt, enter the command **wsadmin.bat** to get a wsadmin command prompt.



Note

If security is enabled or the default SOAP ports have been changed, you will need to pass additional parameters to the batch file in order to get a wsadmin prompt. For example:

```
wsadmin.bat [-host host_name] [-port port_number] [-user userid[-password password]]
```


**Note**

You can connect wsadmin to any of the server JVMs in the cell. After running the wsadmin command it will display the server process for which it has attached to. Depending on the process that it has attached to, you can get thread dumps for various JVMs. If wsadmin is connected to deployment manager, then you can get thread dumps for any JVM in that cell. If it is attached to a node agent, then you can get thread dumps for any JVM in that node. If it is attached to a server, then you can get thread dumps only for the server to which has connected to.

2. Get a handle to the problem application server (Note that the contents in brackets "[.....]", along with the brackets, is not optional. It must be entered to set the jvm object. Also, note that there is a space between *completeObjectName* and *type*.):

```
wsadmin> set jvm [$AdminControl completeObjectName  
type=JVM,process=server1,*]
```

Where *server1* is the name of the application server that does not respond (or is *hung*). If wsadmin is connected to a dmgr and if the server names in cell is not unique, then you can qualify the JVM with node attribute in addition to process.

3. Generate the thread dump:

```
wsadmin>$AdminControl invoke $jvm dumpThreads
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread dump analysis

- Look at the “big picture”
 - A thread dump is a snapshot
 - Take several, giving the JVM time to recover after each
 - They vary in format and detail among platforms
 - Also, it can be difficult to find where each platform writes the dumpfile

- What are the trends in the thread dump?
 - Lots of threads waiting in the same method for some resource
 - Probably a synchronization issue
 - Could be a remote outage
 - No activity
 - WebSphere Application Server is not receiving traffic for some reason
 - Check front-end resources, networks, test clients, and so on
 - Also check timing of the thread dump
 - Hundreds of threads
 - Shared resource not available
 - Customer needs to control Web Container threads better
 - May need more capacity

Figure 7-9. Thread dump analysis

WA5711.0

Notes:

Thread dumps provide a snapshot of the running JVM. They can be useful when CPU utilization is not as expected (lower or higher) or when throughput has hit a plateau.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore hang indicators

- JVM monitor information
 - Shows synchronization locks
 - Indicates blocked threads
 - Look for the string “Deadlock detected”
- Active threads
 - Look for running threads indicated by `state:R`

```

    "Servlet.Engine.Transports:239" (TID:0x34B94018,
    sys_thread_t:0x7CD4E008, state:R, native ID:0x10506) prio=5
    at
    java.net.SocketInputStream.socketRead(Native Method)
    at
    java.net.SocketInputStream.read(SocketInputStream.java (Compiled
    Code))
    at
    com.ibm.ws.io.Stream.read(Stream.java (Compiled Code))
    at
    com.ibm.ws.io.ReadStream.readBuffer(ReadStream.java (Compiled
    Code))
  
```

- This example shows that the thread is performing I/O. If this thread is performing the same operation across multiple javacore files, there may be a network interface issue.

Figure 7-10. Javacore hang indicators

WA5711.0

Notes:

The monitor information shows what synchronization locks are held by which threads. It also shows which threads are blocked by monitors. This information is useful for determining the cause of a deadlocked or hung JVM. The monitor information is in a section entitled LOCKS subcomponent dump routine. It is before the thread dump of all the threads of the JVM.

A large number of threads blocked on a monitor does not mean a deadlock has occurred. It might mean that there is a monitor (synchronization lock) that is causing a backlog of work to be completed.

The javacore processing dumps the current stack for every thread in the JVM. It shows the current state of the thread and produces a stack trace.

- **Thread states:** The thread state indicates if the thread is currently runnable or not. If the thread state is `state:R`, the thread is runnable. If the thread state is `state:CW` (Conditioned Wait) then the thread is in a wait state. If too many of the threads are in the CW state you should focus on the monitor section to look for synchronized hangs.

The values of state can be:

- R - Runnable - the thread is able to run when given the chance.
- CW - Condition Wait - the thread is waiting. For example, because
 - A **sleep()** call is made
 - The thread has been blocked for I/O
 - A synchronized method of an object locked by another thread has been called
 - The thread is synchronizing with another thread with a **join()** call
- S – Suspended – the thread has been suspended by another thread.
- Z – Zombie – the thread has been killed.
- P – Parked – the thread has been parked by the new concurrency API (java.util.concurrent).
- B – Blocked – the thread is waiting to obtain a lock that something else currently owns.
- **Java stack:** The call stack under the thread header is the Java stack. This shows the Java calls that have been made to get the thread to its current state. The first line in the Java stack is the last Java method call that was made. It was from that location that a call into a native method might have been made. That is typically identified with the phrase **Native Method** showing the location in the Java program that was called.
- **Native stack:** The native stack shows what native methods (procedures) were called after the thread entered the native code. The first line in the native stack shows what the thread was doing in native code when the javacore was taken.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore hang symptoms

- Check to see if threads are blocked waiting on monitors
 - May indicate bottleneck on unavailable resources or poor synchronization logic
 - Deadlocks within the process will be noted in the javacore

- If threads are in running state
 - Check method across multiple javacores
 - If individual threads in same method
 - May indicate looping logic

- If threads are in wait states
 - May indicate that a resource is causing hang

Figure 7-11. Javacore hang symptoms

WA5711.0

Notes:

Threads waiting on monitors are not usually deadlocks. Most of these are bottleneck or synchronization issues where the active thread is in some type of long timeout or resource shortage issue.

Javacores contain a lot of information and may cover dozens of threads. It is recommended that tools be used to process the javacore such as the Thread Analyzer. If such a tool is not available, important information can be gained, though the process can be a bit tedious. Most tools will provide the capability to compare javacore files making it easier to check the status of threads over time.

If a hung thread is suspected, also check for log messages from the hang detection facility included with WebSphere.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Hang detection tools

- ThreadMonitor
 - ThreadMonitor architecture was created to monitor thread pools within WebSphere
 - WebSphere 6.x monitored pools include the Web Container thread pool, the ORB thread pool and more
 - Notification of a hang is logged

- ThreadAnalyzer
 - GUI-based tool
 - Gathers and analyzes thread dumps from a WebSphere Application Server
 - Provides recommendations based on analysis

Figure 7-12. Hang detection tools

WA5711.0

Notes:

For the ThreadMonitor, the administrator can determine the threshold of how much time a thread can run before it is considered hung. By default, the monitoring is always on and has a threshold of 10 minutes and a check interval of three minutes. The performance degradation for this monitoring is less than 1%.

These tools will be covered in more details in the following sections.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic summary

Having completed this topic, you should be able to:

- Describe a hang thread condition
- Articulate how to detect hung threads
- Analyze a javacore file to locate hung threads

Figure 7-13. Topic summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Hung thread detection

After completing this topic, you should be able to:

- Describe the features of the thread monitor for hang detection
- Know how to configure and view the output from the WebSphere hang thread detection facility

Figure 7-14. Hung thread detection

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere hung thread detection

- WebSphere contains a built-in hung thread detection function
- ThreadMonitor architecture was created to monitor thread pools within WebSphere
 - The ThreadMonitor monitors Web Container, ORB, and Async Bean thread pools
 - Enabled by default
- Unmanaged threads are not monitored.
 - Threads created by applications (illegal in J2EE)
 - Some internal threads
- Upon notification of a hung thread:
 - Obtain a javacore and see what the thread is doing
 - Investigate the nature of the thread

Figure 7-15. WebSphere hung thread detection

WA5711.0

Notes:

The hang detection option for WebSphere Application Server is turned on by default. You can configure a hang detection policy to accommodate your applications and environment so that potential hangs can be reported, providing earlier detection of failing servers. When a hung thread is detected, WebSphere Application Server notifies you so that you can troubleshoot the problem.

You can then use the javacore or thread dump to see specifically what the identified thread was performing. Investigate the nature of the thread, the nature of the application and the frequency of this kind notice. Decide if this is normal.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Hung thread detection internals

- When the thread pool gives work to a thread, it notifies the thread monitor
 - Thread monitor notes thread ID and timestamp
- Thread monitor compares active threads to timestamps
 - Threads active longer than the time limit are marked “potentially hung”
- Performance impact is minimal (< 1%)

Figure 7-16. Hung thread detection internals

WA5711.0

Notes:

When the thread pool issues work to a thread, it sends a notification to the thread monitor, which notes the thread ID and the time in a list.

At user-configurable intervals, the thread monitor looks at the active threads, and compares them to the list, to determine how long each thread has been active. If a thread has been active longer than the user-specified threshold, the thread is marked as “potentially hung,” and the notifications are sent.

The performance impact of this monitoring is minimal—less than 1%.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Hung thread detection notification

- No action taken to kill the thread; it is only a notification mechanism
- When a thread is suspected to be hung, notification is sent three ways:
 - JMX notification for JMX listeners
 - ThreadPool metric for PMI clients
 - Counters are updated
 - Message written to SystemOut.log:

```
[4/17/04 11:51:30:243 EST] 2d757854 ThreadMonitor W CWWSR0605W:  
Thread Servlet.Engine.Transports : 0 has been active for 14,198  
milliseconds and may be hung. There are 1 threads in total in the  
server that may be hung.
```

Figure 7-17. Hung thread detection notification

WA5711.0

Notes:

The thread monitor does not try to deal with the hung threads, it just issues notifications, so that the administrator or developer can deal with the issues.

When a hung thread is detected, three notifications are sent: a JMX notification for JMX listeners, PMI Thread Pool data is updated for tools like the Tivoli Performance Viewer, and a message is written to the SystemOut log.

JMX Notification

A Java Management Extensions (JMX) notification enables third-party tools to catch the event and take appropriate action, such as triggering a JVM thread dump of the server, or issuing an electronic page or e-mail. The following JMX notification events are defined in the `com.ibm.websphere.management.NotificationConstants` class:

- **TYPE_THREAD_MONITOR_THREAD_HUNG** This event is triggered by the detection of a (potentially) hung thread.

- **TYPE_THREAD_MONITOR_THREAD_CLEAR** This event is triggered if a thread that was previously reported as hung completes its work.

SystemOut.log:

Log format: WSVR0605W: Thread *threadname* has been active for *hangtime* and may be hung. There are *totalthreads* threads in total in the server that may be hung.

- *threadname* is the name that appears in a JVM thread dump
- *hangtime* gives an approximation of how long the thread has been active
- *totalthreads* gives an overall assessment of the system threads.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Hung thread detection false alarms

- What about false alarms?
 - For example, a thread that takes several minutes to complete a long-running query
- If a thread previously reported to be hung completes its work, a notification is sent:

```
[2/17/04 11:51:47:210 EST] 76e0b856 ThreadMonitor W
WSVR0606W: Thread Servlet.Engine.Transports : 0 was
previously reported to be hung but has completed. It was
active for approximately 31,166 milliseconds. There are 0
threads in total in the server that still may be hung.
```
- The monitor has a self-adjusting system to make a best effort to deal with false alarms.

Figure 7-18. Hung thread detection false alarms

WA5711.0

Notes:

It is possible that a thread could actually be running for longer than the specified threshold for legitimate reasons. For example, a thread could be executing a large database query that takes several minutes to return.

The thread monitor is built to recognize false alarms and adjust itself automatically. When a thread that was previously marked as “potentially hung” completes its work and exits, a notification is sent. After a certain number of false alarms, the threshold is automatically increased by 50% to account for these long-running threads. The idea is that if there are several threads that are routinely active for 20 minutes, the threshold will eventually adjust itself to be higher than 20 minutes, so as to not mark those threads as hung.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Hung thread detection configuration

- Create custom properties on the application server
- To configure:
 - Navigate to **Servers -> Application Servers -> server_name**
 - Under Server Infrastructure, click **Administration -> Custom Properties**
 - Add the following properties (or change if present)

Property	Units	Default	Description
com.ibm.websphere.threadmonitor.interval	secs.	180	Interval at which the thread pools will be polled for hung threads
com.ibm.websphere.threadmonitor.threshold	secs.	600	The length of time that a thread can be active before being marked as "potentially hung"
com.ibm.websphere.threadmonitor.false.alarm.threshold	N/A	100	The number of false alarms that can occur before automatically increasing the threshold by 50%.

Figure 7-19. Hung thread detection configuration

WA5711.0

Notes:

These custom properties will not be present unless added to the configuration. If not present, the default values take effect. To change the behavior add these properties providing desired values.

The hang detection policy can be configured by creating custom properties for the application server.

- **com.ibm.threadmonitor.interval** is the interval at which the thread pools will be polled for hung threads (in seconds). It defaults to 180 seconds, which is 3 minutes.
- **com.ibm.websphere.threadmonitor.threshold** is the length of time that a thread can be active before being marked as "potentially hung." The default value is ten minutes.
- **com.ibm.websphere.threadmonitor.false.alarm.threshold** is the number of false alarms that can occur before automatically increasing the threshold by 50%. The default value is 100. Automatic adjustment can be disabled altogether by setting this property to zero.

The application server must be restarted for these changes to take effect. To adjust the hang detection policy on the fly, use wsadmin. Refer to the information center for instructions.

To disable the hang detection option, set the **com.ibm.websphere.threadmonitor.interval** property to less than or equal to zero.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic summary

Having completed this topic, you should be able to:

- Describe the features of the thread monitor for hang detection
- Know how to configure and view the output from the WebSphere hang thread detection facility

Figure 7-20. Topic summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer

After completing this topic, you should be able to:

- Describe the features and interface of Thread Analyzer
- Describe the uses in problem determination of Thread Analyzer

Figure 7-21. Thread Analyzer

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere Thread Analyzer

- A tech-preview tool to analyze thread dumps
 - Available from the IBM Support Assistant
- Useful for analyzing synch points in large dumps or comparing dumps
- Friendlier interface for novice thread dump readers
 - Provides graphical interface to view contents of the thread dump
- Analyzes WebSphere Application Server thread dumps and provides recommendations based on the analysis
- Use to analyze threads for the following
 - Performance bottlenecks due to either WebSphere configuration or application problems
 - Determining if deadlocks are being created
 - Determining if threads are being blocked on monitors (may not be a deadlock)

Figure 7-22. WebSphere Thread Analyzer

WA5711.0

Notes:

Thread Analyzer is supported on Solaris, AIX, Windows, and Linux platforms.

By using the top of stacks (TOS) analysis found in the Overall Thread Analysis section, you can use Thread Analyzer to pin-point areas of high contention in your code.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer functions

- Analysis section
 - Breakdown of where current threads are executing in JVM
 - Use to determine where investigation into thread dump should start
- Overall Monitor Analysis section
 - Provides analysis of monitors within JVM
 - Use to determine if deadlocks are to blame for problem
 - Shows classic deadlocks and single-threaded waiters that exist during the execution of your application
- Overall Thread Analysis section
 - Provides top of stack (TOS) analysis for threads in JVM
 - Use to determine areas of high contention in your application
 - Look for methods that are being executed by majority of threads in JVM
- Remaining sections offer similar information but break it down by container and offer advice in just those areas
 - Advice specifically for Web container and ORB service

Figure 7-23. Thread Analyzer functions

WA5711.0

Notes:

The Thread Analyzer provides the capability to compare thread dumps and it is possible to filter the information displayed.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer: Summary

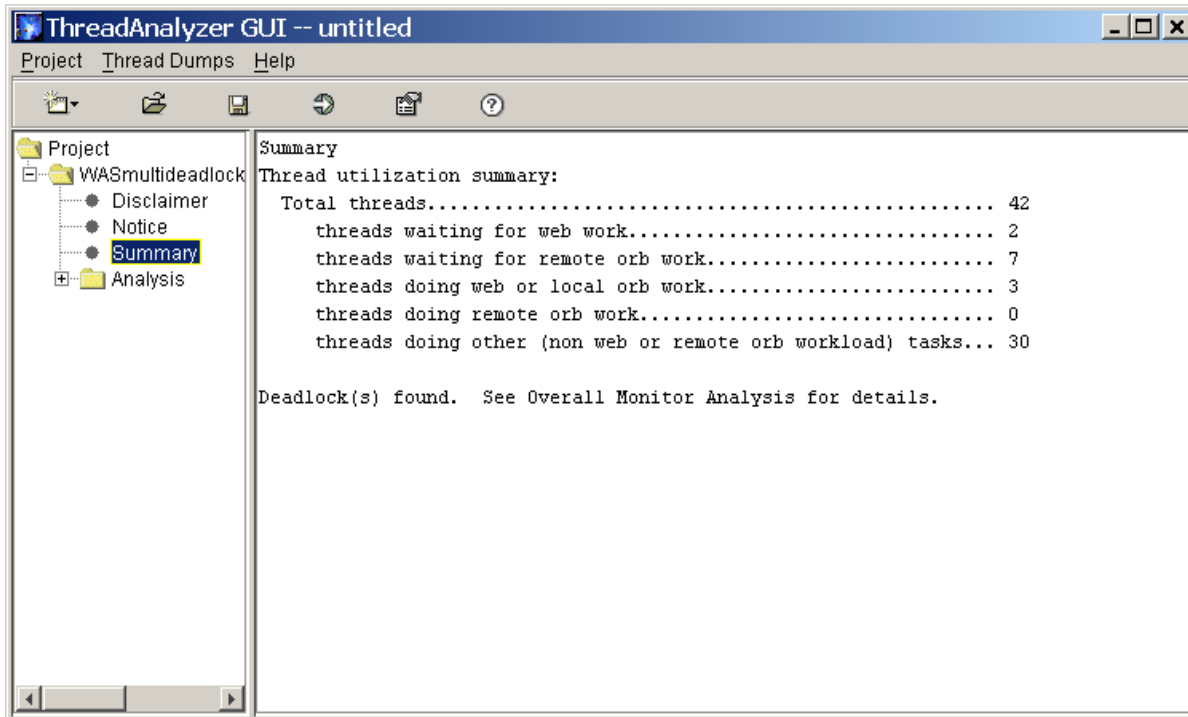


Figure 7-24. Thread Analyzer: Summary

WA5711.0

Notes:

The Summary view displays the breakdown of where threads are executing in the JVM. It also provides a quick point of reference to determine if there are any deadlocks present in the thread dump.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer: Analysis

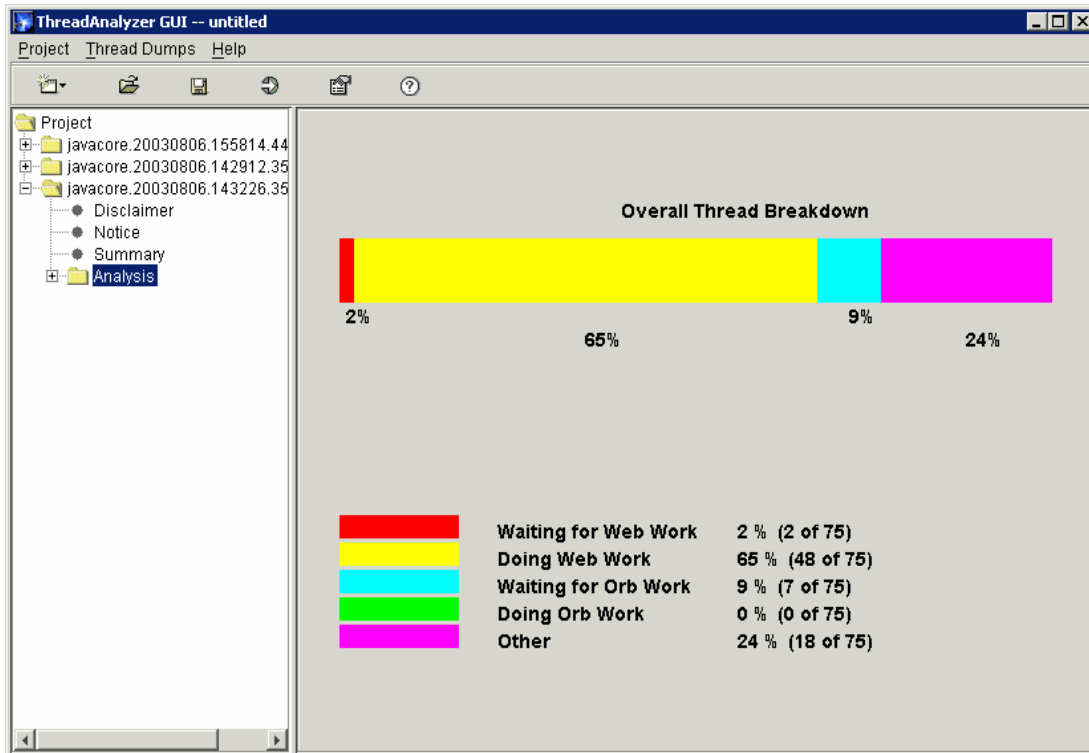


Figure 7-25. Thread Analyzer: Analysis

WA5711.0

Notes:

This screen capture shows the overall breakdown graphic of the threads in the snapshot. The graph shows what all of the current threads in the JVM are doing.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer: Multiple dump analysis

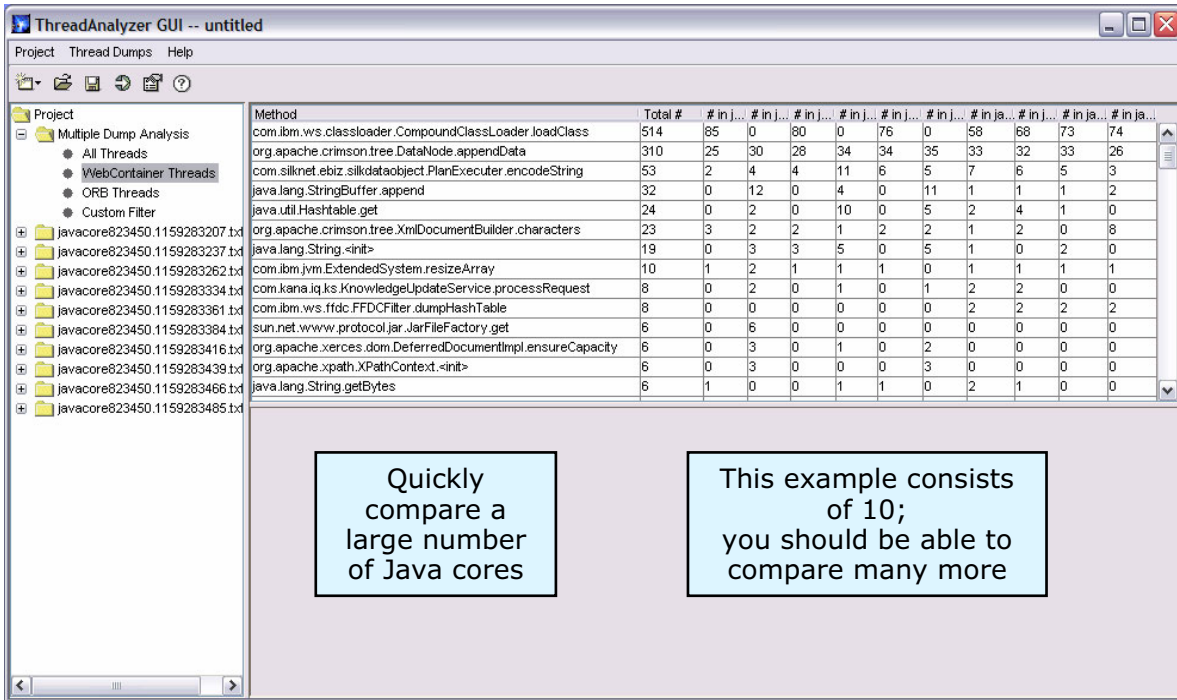


Figure 7-26. Thread Analyzer: Multiple dump analysis

WA5711.0

Notes:

Multiple dump analysis is automatically available after opening a series of dumps. The opened dumps must be related in order for the data to make sense.

This view can show how thread behavior changes across a sequence of thread dumps. Look to see if threads are moving or if over time more threads are being consumed by a specific method.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer: Overall thread analysis

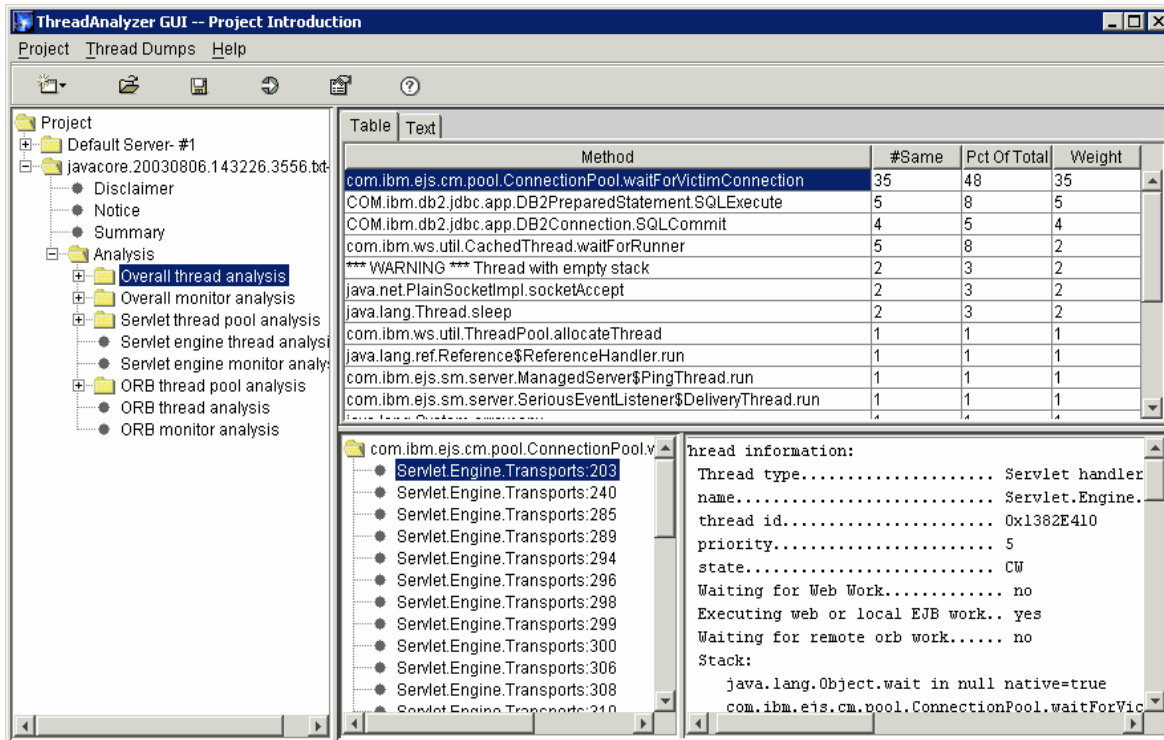


Figure 7-27. Thread Analyzer: Overall thread analysis

WA5711.0

Notes:

In the screen capture above:

- The left panel allows you to access the different open projects and thread dumps. Right now the javacore.20030806.143226.3556.txt thread dump is open. To see the raw thread dump, click this top level. From this panel you can select any of the different analysis information (Summary, Analysis, Overall thread analysis, and so on) for the thread dump.
- The top panel on the right side of the screen capture displays the details from the Overall thread analysis section. In this case the information shows all of the methods that were being executed and the number of threads that were executing them. You can also look at the raw text data as well by flipping to the **Text** tab.
- The bottom-left panel on the right side of the screen capture displays the details from selecting the **com.ibm.ejs.cm.pool.ConnectionPool.waitForVictimConnection** method in the panel above. The panel displays the exact threads that were executing the method at the time of the thread dump.

- The bottom-right panel on the right side of the screen capture displays information for the selected thread (ServletEngine.Transports:203). A description of the State information follows:

R (Runnable) - The thread is able to run when given the chance.

CW (Condition Wait) - The thread is waiting. For example, because:

- A **sleep()** call is made.
- The thread has been blocked for I/O.
- A synchronized method of an object locked by another thread has been called.
- The thread is synchronizing with another thread with a **join()** call.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer: Deadlock indicators

- Folder icon will indicate if a deadlock was detected

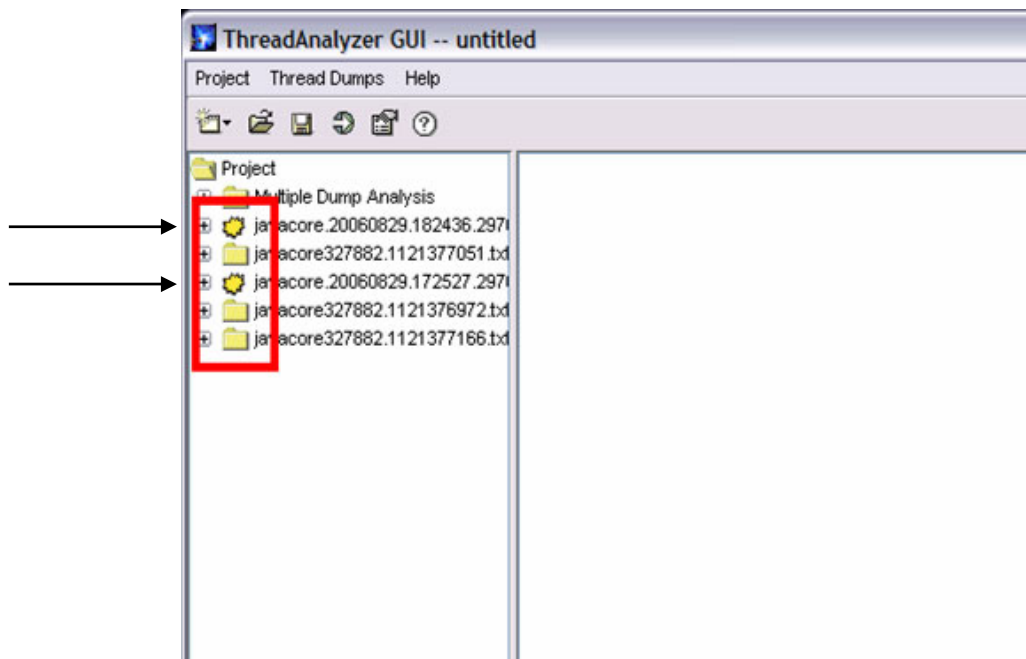


Figure 7-28. Thread Analyzer: Deadlock indicators

WA5711.0

Notes:

The icon for the javacore folder changes to alert a deadlock has occurred. The screen capture above illustrates this: the first and third snapshots were found to contain deadlocks.

To see detailed information on the deadlock that was detected, expand the javacore folder and select **Analysis -> Overall monitor analysis**.

Instructor notes:

Purpose —

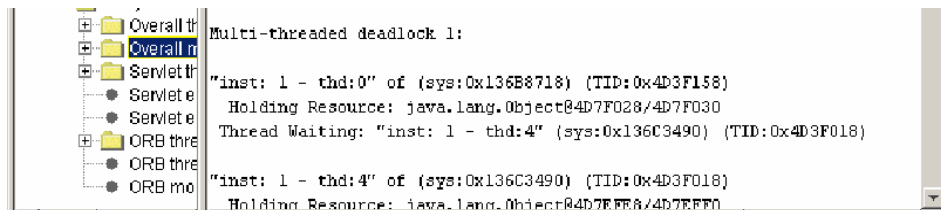
Details —

Additional information —

Transition statement —

Thread Analyzer: Overall monitor analysis

- Classic deadlock analysis



- Single-threaded waiter analysis

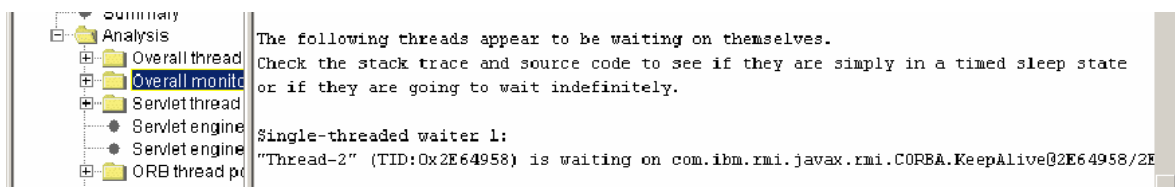


Figure 7-29. Thread Analyzer: Overall monitor analysis

WA5711.0

Notes:

The Overall monitor analysis section displays details on any deadlocks present in the thread dump. The first part of the analysis has information pertaining to classic deadlocks, whereas the second part of the analysis has information pertaining to single-threaded waiters.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Example output: Servlet thread analysis

```
Servlet engine thread analysis
COM.ibm.db2.jdbc.app.DB2PreparedStatement.SQLExecute
seems to be currently executing on 40 servlet threads.
Since 78% (40 out of 51) of the threads doing servlet work seem to be
executing this method, it would seem that there
is some possibility that this method and its call path
may warrant investigation.
Servlets affected:
trade_client.TradeScenarioServlet [40 occurrences]
Callers (servlet threads only):
trade.TradeBean.getBalance [1]
trade.TradeBean.buy [1]
COM.ibm.db2.jdbc.app.DB2PreparedStatement.executeQuery [11]
COM.ibm.db2.jdbc.app.DB2PreparedStatement.executeUpdate [11]
trade.EJSJDBCPersisterCMPAccountBean._create [2]
trade.EJSJDBCPersisterCMPHoldingBean.findByUserID [2]
trade.EJSJDBCPersisterCMPQuoteBean.load [1]
trade.EJSJDBCPersisterCMPProfileBean.load [2]
trade.EJSRemoteStatelessTrade.login [2]
trade.EJSJDBCPersisterCMPRegistryBean.store [2]
trade.EJSJDBCPersisterCMPAccountBean.load [1]
```

Figure 7-30. Example output: Servlet thread analysis

WA5711.0

Notes:

Circled above is an example of the type of recommendations that you will receive from Thread Analyzer. The message shows how the tool tries to point you in the initial direction of a problem area.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Thread Analyzer: Custom filters

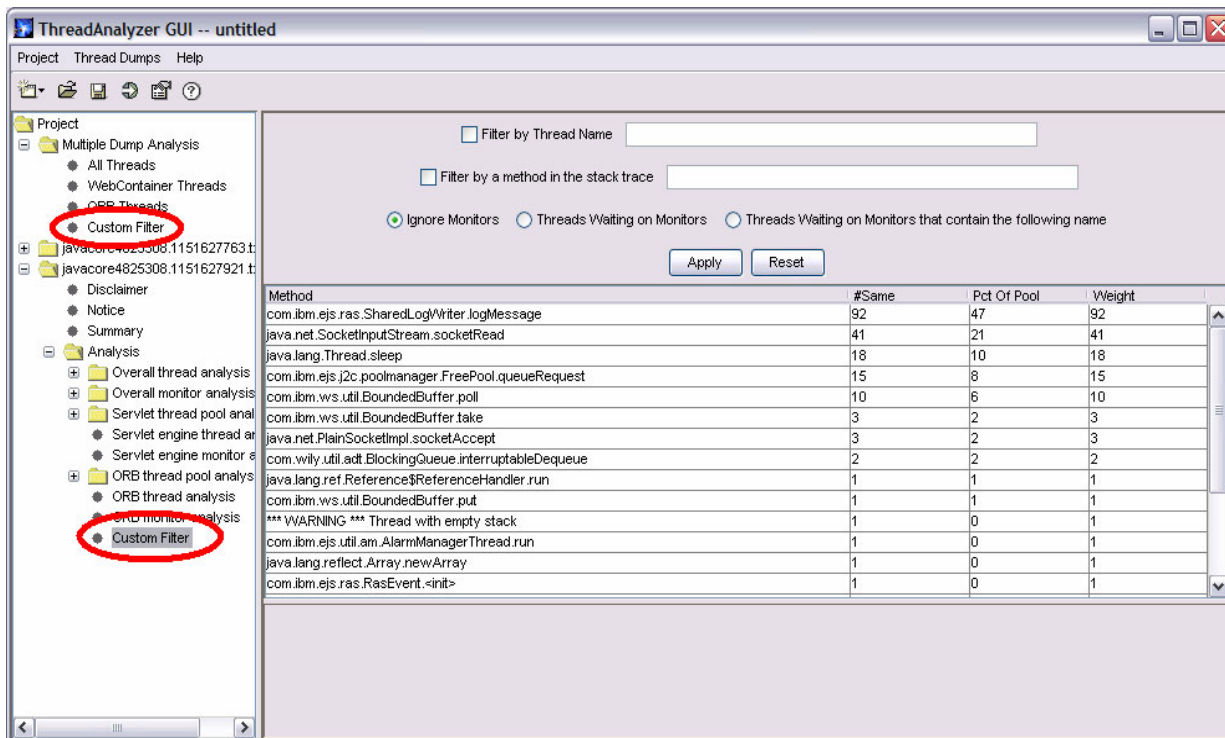


Figure 7-31. Thread Analyzer: Custom filters

WA5711.0

Notes:

Custom filters are used for threads or methods that are relevant to the problem you are experiencing. You can filter by thread name, for instance, “Servlet” or by the method name in the stack trace.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic summary

Having completed this topic, you should be able to:

- Describe the features and interface of Thread Analyzer
- Describe the uses in problem determination of Thread Analyzer

Figure 7-32. Topic summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe what a hang is and be able to detect one
- Analyze javacore files for hangs
- Use the WebSphere Application Server hang detection facility
- Use the ThreadAnalyzer

Figure 7-33. Unit summary

WA5711.0

Notes:

This concludes this unit.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 8. How to troubleshoot crashes

Estimated time

00:30

What this unit is about

This unit looks at how to detect and troubleshoot a crash.

What you should be able to do

After completing this unit, you should be able to:

- Describes what a crash is
- Detect a crash
- Analyze a javacore file for a crash
- Analyze system core files
- Discuss the tools available for troubleshooting a crash

How you will check your progress

Accountability:

- Machine exercises

References

SG24-6798-00 WAS V6 Problem Determination for Distributed Platforms

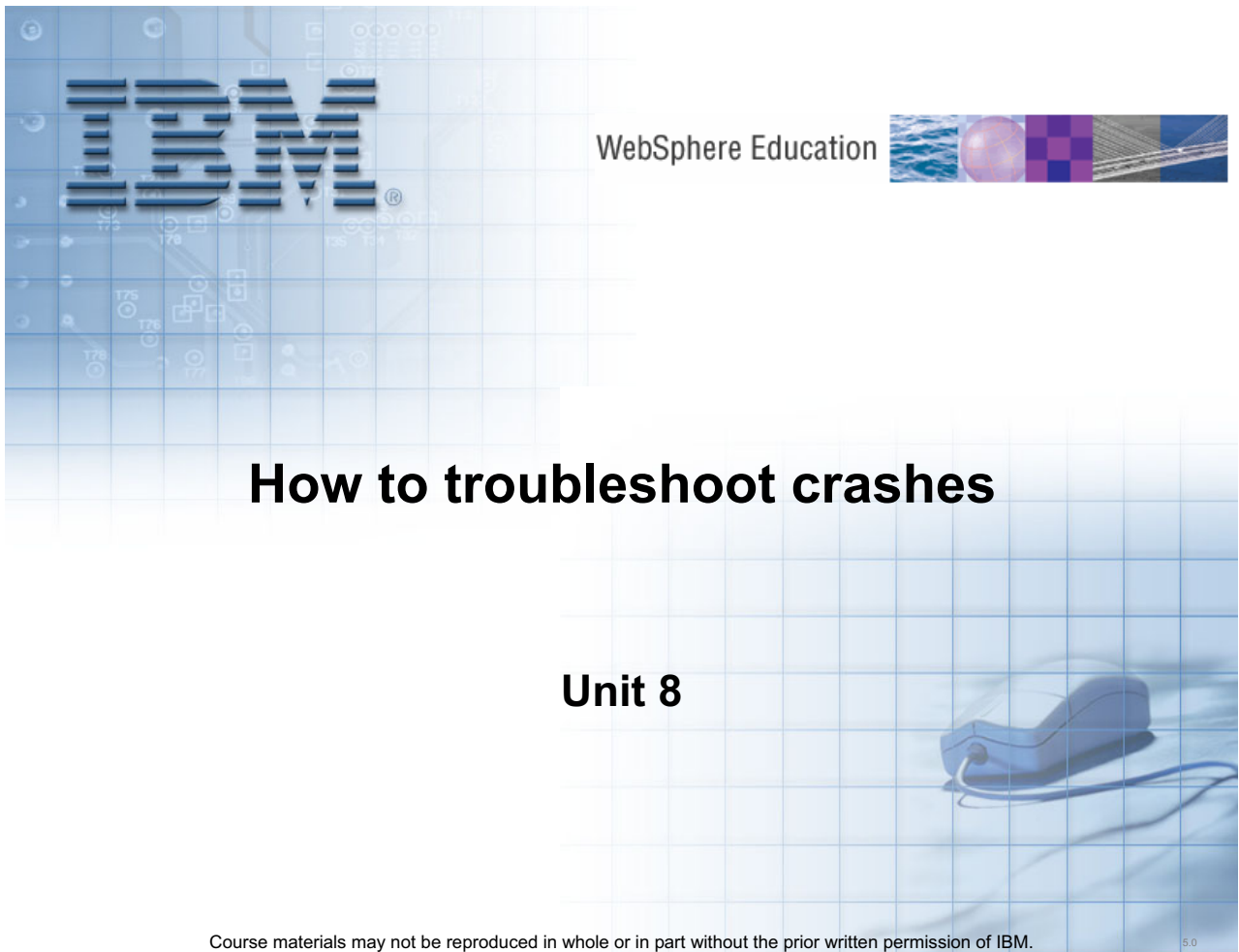


Figure 8-1. How to troubleshoot crashes

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe what a crash is and be able to detect one
- Analyze javacore files for crash
- Use the Dump Analyzer when javacore files are not available

Figure 8-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Debug a crash

After completing this topic, you should be able to:

- Describe a JVM crash
- Describe how to control what dump data is generated
 - List common operating system signals related to a crash
- Analyze a javacore file to detect crash symptoms

Figure 8-3. Debug a crash

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM process crash defined

- Not the same as thread hang

- Symptoms
 - Process terminated with Java exception or native signal

- Usual causes
 - Out of memory exception
 - Call stack overflow
 - Unexpected exception (for example, out of disk space)
 - Optimizer failure (for example, JiT)
 - Bad JNI call or library problem
 - Segmentation violations while executing native code

Figure 8-4. JVM process crash defined

WA5711.0

Notes:

All JVMs use the just-in-time (JIT) compiler to compile heavily used Java bytecode into native instructions during server run time to enhance performance.

Crashes can be categorized into three main types:

- Crashes caused by some sort of problem in the environment where the JVM executes, such as a lack of resources: `OutOfMemory`, `StackOverflow`, unexpected exception, and so on. These will be fixed by fixing the application or the environment in which the problem occurred.
- Crashes caused by an internal error (a defect) in the JVM implementation (for example, JIT). These will be fixed by applying a JDK patch.
- Crashes caused by an internal error in some external third-party JNI library that the customer uses and has loaded in the JVM. These will be fixed by applying a patch to the third-party library, or avoiding the use of that particular library.

For crashes due to things like `OutOfMemory` and `StackOverflow`, the JVM usually gets a chance to do some cleanup before exiting, and in particular, it usually will produce a

javacore file. On the other hand, with internal JVM errors and JNI errors, you often get a “hard” crash, with just a core file but no javacore file.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Crash problem determination: Data to collect

- Core files
 - Also known as process dumps or system core files
 - Complete dump of the virtual memory in binary format
 - Can be quite large
 - Tools available to parse files into readable formats

- Javacore files
 - Also known as javadump or thread dump files
 - Text file created by an application server during a failure
 - Can also be triggered manually
 - Error condition will be given at top of dump

- Steps to collect data
 - Look for a javacore file automatically created during a crash
 - If one was not created, reproduce the problem and try to manually create one just before the crash occurs
 - If you are unable to generate a javacore file, creating a system core file may be your only option

Figure 8-5. Crash problem determination: Data to collect

WA5711.0

Notes:

The JVM supports the ability to generate a native system dump, also known as process dump. Process dumps are a snapshot of the contents of the entire process in binary format. Information about what was going on in that process when the dump was taken can be extracted and presented in a readable format similar to a javacore.

The javacore files are text files that contains only a summary of the information that is most pertinent to the most common problems.

Javacores are enabled by default, can be disabled by setting the `DISABLE_JAVADUMP` environment variable to `TRUE`. Javacore files can be created by both IBM and Sun JDKs.

On Sun/HP platforms, verbose thread dump information is written to `native_stdout.log`.

Process dump file locations:

- AIX: Output is written to a core file named `core.&processid.×tamp.txt` that is in the current working directory.
- Windows: Output is written to a file named `core.&processid.×tamp.dmp` into the same directory that is used for `JAVADUMP`.

- Linux: Process dumps are not available

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

-Xdump JVM command line argument

- New to JDK V5, the **-Xdump** option controls the way you use dump agents to create dumps (types include system, Java and heap)
- The **-Xdump** option allows you to:
 - Add and remove dump agents for various JVM events
 - Update default dump agent settings
 - Limit the number of dumps produced
 - Show dump agent help
- Preferred way to control dump agent behavior moving forward (JDK v5 and on).
 - Use of variables to affect dump behavior is still supported (although use is being deprecated)
- For detailed description of **-Xdump** usage:
 - <http://www.ibm.com/support/docview.wss?uid=swg21242497>

Figure 8-6. -Xdump JVM command line argument

WA5711.0

Notes:

To display on JVM startup the conditions (if any) that will generate a heapdump (or javadump or systemdump), you can use **-Xdump:what**.

You can have multiple **-Xdump** options on the command line and also multiple dump types triggered by multiple events.

For example:

```
java -Xdump:heap:none -Xdump:heap+java:events=vmstart+vmstop <class>  
[args...]
```

would turn off all Heapdumps and create a dump agent that produces a Heapdump and a Javadump when either a vmstart or vmstop event occurs.

The dump agent processing ensures that multiple **-Xdump** options are optimized. You can use the **-Xdump:what** option to list the registered dump agents. The events keyword is used as the prime trigger mechanism. However, you can use additional keywords to further control the dump produced.

**Hint**

For a complete description of **-Xdump** behavior and options, read the JDK 5.0 Diagnostic Guide.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM dump initiation: Events

- Events (or conditions) that can cause dumps:

EXCEPTION	Unexpected synchronous terminating signal; that is, unrecoverable storage violation.
ERROR	Controlled abort due to an error detected internally; for example, no more memory is available.
INTERRUPT	Asynchronous terminating signal; for example, you pressed Ctrl+C.
DUMP	This can be caused if you press Ctrl+Break on Windows, Ctrl+V on z/OS, or Ctrl+\ on AIX or Linux.
OUTOFMEMORY	The JVM cannot satisfy a request for storage.

Figure 8-7. JVM dump initiation: Events

WA5711.0

Notes:

The JVM might produce dump files in response to specific events, depending on the setting of the environment variables `JAVA_DUMP_OPTS` and `JAVA_DUMP_TOOL`.

By default `INTERRUPT` events will not initiate a dump as opposed to `EXCEPTIONS` and `DUMP` events which will cause a dump to be generated.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM dump initiation: Types

- The types of dump that can be produced are:

SYSDUMP	An unformatted dump that the operating system generated (basically a core file).
User specified	Whatever the JAVA_DUMP_TOOL variable specifies.
HEAPDUMP	An internally-generated dump of the objects that are on the Java heap.
JAVADUMP	An internally-generated and formatted analysis of the JVM.

- Note that platform specific variations exist

Figure 8-8. JVM dump initiation: Types

WA5711.0

Notes:

If the JAVA_DUMP_TOOL environment variable is set, that variable is assumed to specify a valid executable name.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JAVA_DUMP_OPTS variable (use now deprecated)

- Which dumps are produced for which condition is determined by the JAVA_DUMP_OPTS variable as follows:

```
JAVA_DUMP_OPTS="ONcondition(dumptype[count], dumptype[count]),  
ONcondition(dumptype[count],...)"
```

- *condition* can be: ANYSIGNAL, DUMP, ERROR, INTERRUPT, EXCEPTION and OUTFOMEMORY
 - *dumptype* can be: ALL, NONE, JAVADUMP, SYSDUMP and HEAPDUMP
 - *count* is the maximum number of dumps of this type to produce.
- For example:
 - ONERROR (SYSDUMP) creates system dumps for error conditions

Figure 8-9. JAVA_DUMP_OPTS variable (use now deprecated)

WA5711.0

Notes:

If you set the JAVA_DUMP_OPTS environment variable, default dump agents will be disabled.

JAVA_DUMP_OPTS is parsed by taking the first (leftmost) occurrence of each condition, so duplicates are ignored. For example: ONERROR(SYSDUMP),ONERROR(JAVADUMP) creates system dumps for error conditions

Be careful in setting JAVA_DUMP_OPTS values as there are also operating system-level specific settings. Incorrect settings may result in a truncated and useless dump. This is a fairly common source of problems for customers. It is highly recommended that you look at the JDK Diagnostic Guide for platform specific variations.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

UNIX operating system common signals

- **SIGQUIT**

- Indicates a command was issued to generate a thread dump
- Typically does not end the JVM process

- **SIGILL**

- Means an illegal instruction was executed
 - This can mean a corruption of the code segment or a branch that is not valid within the native code
- This signal often indicates a problem caused by JIT-compiled code

- **SIGSEGV**

- Indicates an operation that is not valid in a program
 - Example: Accessing an illegal memory address
- This is typically indicative of a programming problem in one of the native libraries

Figure 8-10. UNIX operating system common signals

WA5711.0

Notes:

Keep in mind that non-JVM processes (that is, non Java-based applications) will end if a quit signal is received.

The javacore file displays the operating system signal that caused the javacore file to be written. Some signals also produce a core file in UNIX operating systems. The JVM signal handler library is libhpi.a.

Following are some of the most commonly seen signals in javacore files:

- **SIGQUIT:** This signal is sent when a **kill -3** command is issued against the JVM process. This signal typically does not end the JVM process. It generates a thread dump (javacore) for diagnosing a potentially hung JVM process.
- **SIGILL:** This is the equivalent of a **kill -4** command. This means an illegal instruction was executed. This can mean a corruption of the code segment or an branch that is not valid within the native code. This signal often indicates a problem caused by JIT-compiled code.

- **SIGSEGV**: This signal is equivalent to a **kill -11** command. It indicates an operation that is not valid in a program, such as accessing an illegal memory address. This is typically indicative of a programming problem in one of the native libraries.

Signals -1,0, **OUTOFMEMORY** and **SIGNONE** are the memory signals. These will be discussed in a later unit.

Signals 10, 11, **SIGBUS** and **SIGSEGV** indicate an application server crash.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Windows operating system common signals

- **Memory access error**
 - Invalid memory address
 - JVM action: javacore file and abort

- **Illegal access error**
 - JVM action: javacore file and abort

Figure 8-11. Windows operating system common signals

WA5711.0

Notes:

The signals that Windows uses are entirely different than those used by UNIX. Here is a list of some of the operating system signals produced on Windows. Some signals also produce a user.dmp file. The JVM signal handler library is hpi.dll.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore subcomponents helpful for crash debug

TITLE	Shows basic information about the event that caused the generation of the Javadump, the time it was taken, and the file name.
GPINFO	Shows some general information about the operating system. If the dump was caused by a general protection fault (GPF), information about the failure is provided. Namely the fault module is identified.
ENVINFO	Shows information about the JRE level, details about the command line that launched the JVM process and the JVM environment.
THREADS	Identifies the current thread and provides a stack trace.

Figure 8-12. Javacore subcomponents helpful for crash debug

WA5711.0

Notes:

It is important to understand what signal caused the javacore. This determines how to interpret the information in the javacore file. Does the signal indicate a JVM crash? Does the signal information state it was due to an operator action (noted as “user” dump event)? If the javacore is written to diagnose a hung JVM process, the steps you take are very different than if the javacore is due to a JVM crash

Use the Java command line to determine if the javacore is for a WebSphere Application Server Java process. If so, which WebSphere Java process it is for: an Application Server process, an Administrative Server process, a jmsserver, a node agent, or some other WebSphere Application Server process.

The current thread is the thread that is running when the signal is raised that causes the javacore to be written. The **Current Thread Details** shows the Java and native stacks of the current thread. Since these are stacks, the most recent calls are at the top of the stack.

- The Java stack shows the Java method calls that are made by the current thread

- If there is a native stack, this contains the most recent events that the thread executed. As the name implies, these are events in native code (libraries) called by Java

This can be used to determine what was executing when the crash occurred.

To verify that the javacore file is complete you should see a “END OF DUMP” string at the end of the file.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore example showing a crash

```

NULL -----
0SECTION      TITLE subcomponent dump routine
NULL =====
1TISIGINFO    Dump Event "gpf" (00002000) received
1TIDATETIME   Date:                2007/09/25 at 15:26:44
1TIFILENAME   Javacore filename: C:\dev\jnitest\javacore.20070925.152644.11800.txt
NULL -----
0SECTION      GPINFO subcomponent dump routine
NULL =====
2XHOSLEVEL    OS Level           : Windows XP 5.1 build 2600 Service Pa
2XHCPU        Processors      -
3XHCPUARCH    Architecture   : x86
3XHNUMCPUS    How Many       : 2
1XHEXCPMODULE Module: C:\WINDOWS\system32\msvcrt.dll
1XHEXCPMODULE Module_base_address: 77C10000
1XHEXCPMODULE Offset_in_DLL: 000378C0

.....
{deleted lines}

.....
0SECTION      THREADS subcomponent dump routine
NULL =====
NULL
1XMCURTHDINFO Current Thread Details
NULL -----
3XMTHREADINFO "Thread-1514" (TID:0x57BAD300, sys_thread_t:0x429853AC, state:R, native
ID:0x000037D0) prio=5
4XESTACKTRACE at com/ibm/wa571/test/JniTest.setMessages(Native Method)
4XESTACKTRACE at com/ibm/wa571/test/JniTest.run(JniTest.java:115(Compiled Code))
4XESTACKTRACE at java/lang/Thread.run(Thread.java:799(Compiled Code))
NULL

```

Figure 8-13. Javacore example showing a crash

WA5711.0

Notes:

GPF stands for general protection fault.

For example, the above indicates that the javacore file was created as the result of a fault in the C++ run time library (Microsoft Visual C++ run time library). This would indicate a problem related to code involving the Java Native Interface (JNI).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore fault module

- 1XHEXCPMODULE
 - Indicates the module that caused the fault

- Fault module identification:
 - The JVM module:
 - Windows: JVM.dll
 - AIX: libjvm.a
 - Linux: libjvm.so

 - The JIT module:
 - Windows: JITC.dll
 - AIX: libjitc.a
 - Linux: libjitc.so

 - Other modules may be indicated, such as DB2

Figure 8-14. Javacore fault module

WA5711.0

Notes:

Look for the full path name to that library or module as that may indicate the use of some third-party JNI library. It can also point to the use some non-standard library that has been loaded to replace the library that is normally furnished with the JVM.

If the fault module does not identify the library that caused the crash, you must examine the Current Thread Details to see if you can use determine the library that caused the crash.

If either the JVM or JIT fault modules are indicated, first upgrade the JDK and retry to see if the problem goes away. If the problem still exists after the upgrade check for a stack overflow problem in the case of the JVM module or try to work around JIT-related problems.

You might not be able to fix a JIT-related problem. So, the key is to find a workaround that is acceptable while you report the problem to IBM and get a solution. The best way to find a workaround is to determine the method on which the failure is occurring and have the JIT compiler skip this method. An alternative is to completely disable JIT, though this action can have performance implications.

To see if the JIT is crashing in the middle of a compilation, use the **verbose** option with the following additional settings: **-Xjit:verbose={compileStart|compileEnd}**. Also check `native_stderr.log` as it may identify which method is causing the problem.

These verbose settings report when the JIT starts to compile a method, and when it ends. If the JIT fails on a particular method (that is, it starts compiling, but crashes before it can end), use the **exclude=** parameter to exclude it from compilation. If excluding the method prevents the crash, you have an excellent workaround that you can use while the service team corrects your problem.

You can exclude it from compilation altogether, using the **exclude=<method>** parameter: **-Xjit:exclude={methodname}**. For example:

```
-Xjit:exclude={java/lang/Math.max(II) I}
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Javacore current thread details (JDK 1.4.2)

- Examine the current thread details to see which library the current thread was processing at the time of the JVM crash
- Example showing error in JIT

```

1XHCURRENTTHD   Current Thread Details
NULL            -----
2XHCURRSYSTHD   "EntigoAppsStarter" sys_thread_t:0x59AF8650
3XHNATIVESTACK  Native Stack
NULL            -----
3XHSTACKLINE    at 0xD2782A88 in dataflow_arraycheck
3XHSTACKLINE    at 0xD27226A0 in bytecode_optimization_driver
3XHSTACKLINE    at 0xD27251CC in bytecode_optimization
3XHSTACKLINE    at 0xD2685140 in JITGenNativeCode
3XHSTACKLINE    at 0xD26AB774 in jit_compile_a_method_locked
3XHSTACKLINE    at 0xD26ACD24 in jit_compiler_entry
3XHSTACKLINE    at 0xD26AD284 in _jit_fast_compile

```

Figure 8-15. Javacore current thread details (JDK 1.4.2)

WA5711.0

Notes:

If the library is not identified by the signal information, you must examine the current thread details to see if the native stack indicates in which library the current thread was processing at the time of the JVM crash. If the signal is one that is used for an abnormal process termination, that is, SIGSEGV or SIGILL, the current thread details show the sequence of calls that caused the signal to be generated. For a JVM crash, if the line that identifies the signal is not able to identify the failing library, the native stack trace in the current thread details might be useful to identify the library.

In this example, the current thread is in the libjtc.a library, which is generating native code for a Java method. This is not obvious from the first three lines of the native stack, but the fourth call down shows that the JIT compiler is involved. In this case, a workaround is to skip JIT compiling of this method. A likely resolution is to upgrade the Java SDK to upgrade the libjtc.a library.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Steps if crash cause not identified

- Frequently, the javacore file does not clearly identify the cause of the signal. Often the Native Stack will show the following:

```
----- Native Stack -----  
unable to backtrace through native code - iar 0x3062e73c not in text area  
(sp is 0x2ff21748)
```

- Steps you can take:
 - Disable JIT compilation.
 - Upgrading to a more recent JDK can sometimes resolve a problem.
 - Use the core file (on UNIX) or user.dmp file (on Windows) to see if this provides more information.
 - Sometimes a bad Java SDK installation can cause problems

Figure 8-16. Steps if crash cause not identified

WA5711.0

Notes:

Frequently when the crash is not identified in the javacore file, it is due to a problem with the JIT compiled code. Disabling JIT compilation can be used to confirm if this is the case. The JIT compiler is a JVM optimization. There can be significant performance degradation when the JIT is disabled—as much as 20% or more.

If the problem is in the JIT compiler, you might be able to skip JIT compiling for a method if the problem is in the code generated for a method or in the code generation itself.

Sometimes a bad Java SDK installation can cause problems. The fullversion for the javacore file comes from the libjvm.a. The fullversion for the Java **-fullversion** command is from the Java executable. If there is a discrepancy between these two dates, either the wrong libjvm is picked up due to an error in the library path statement, or there is a problem in the JVM installation.

If you did not get a javacore file, check to see if a process dump (core file) exists. If it does, it is still likely that you have had an application server crash.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic summary

Having completed this topic, you should be able to:

- Describe a JVM crash
- Describe how to control what dump data is generated
 - List common operating system signals related to a crash
- Analyze a javacore file to detect crash symptoms

Figure 8-17. Topic summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM Dump Analyzer for Java

After completing this topic, you should be able to:

- Describe the Diagnostic Tooling Framework for Java (DTFJ)
- Know how to process a core dump using the Dump Analyzer for Java

Figure 8-18. IBM Dump Analyzer for Java

WA5711.0

Notes:

The IBM Dump Analyzer for Java is more commonly referred to as the Dump Analyzer.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

What is DTFJ?

- DTFJ (Diagnostic Tooling Framework for Java) is a new technology within the IBM JDK to analyze and diagnose problems in Java applications
 - Read RAS artifacts from a JVM (for example, a core file) and extract all kinds of useful information from that dump
 - Not just one tool: An extensible framework for building many different tools
- Components of the DTFJ family
 - jextract: A tool to capture information from a JVM system dump (for example, core file) and package it into a platform-independent format
 - DTFJ library proper or core library: A library that parses the contents of the system dump file packaged by jextract, and provides access to its contents in a standardized manner, through a standard API
 - DTFJ-based tools: A collection of tools that call the DTFJ library through the DTFJ API, to present and analyze information in various ways useful to the users
- By providing common tooling, the use of specific tools for specific JVM artifacts is avoided

Figure 8-19. What is DTFJ?

WA5711.0

Notes:

The Diagnostic Toolkit and Framework for Java (DTFJ) API is a Java-based API for accessing postmortem information from the system dump of a Java process. This allows tool writers to access information from the dump about the system, the process, the Java VM, and the Java application without having to understand how the relevant structures are laid out in memory. This makes the ability to write postmortem tooling far more accessible.

There are many different RAS artefacts such system dumps (core files, MiniDumps), heap dumps (.phd files), javacore files (javacore...txt files) but no common tooling. Many good reasons for different file formats but often tooling is too JVM software-specific (jformat, jcore, kca).

The DTFJ-based Dump Analyzer tool was created to convert system dump file into human readable format.

In the past there have been various other ad-hoc tools, and sometimes people even use a simple low-level debugger (for example, dbx, gdb, and so on) to try to understand a crash. That is not as good as running a DTFJ tool, but sometimes that is all you can do.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using the DTFJ components - Example

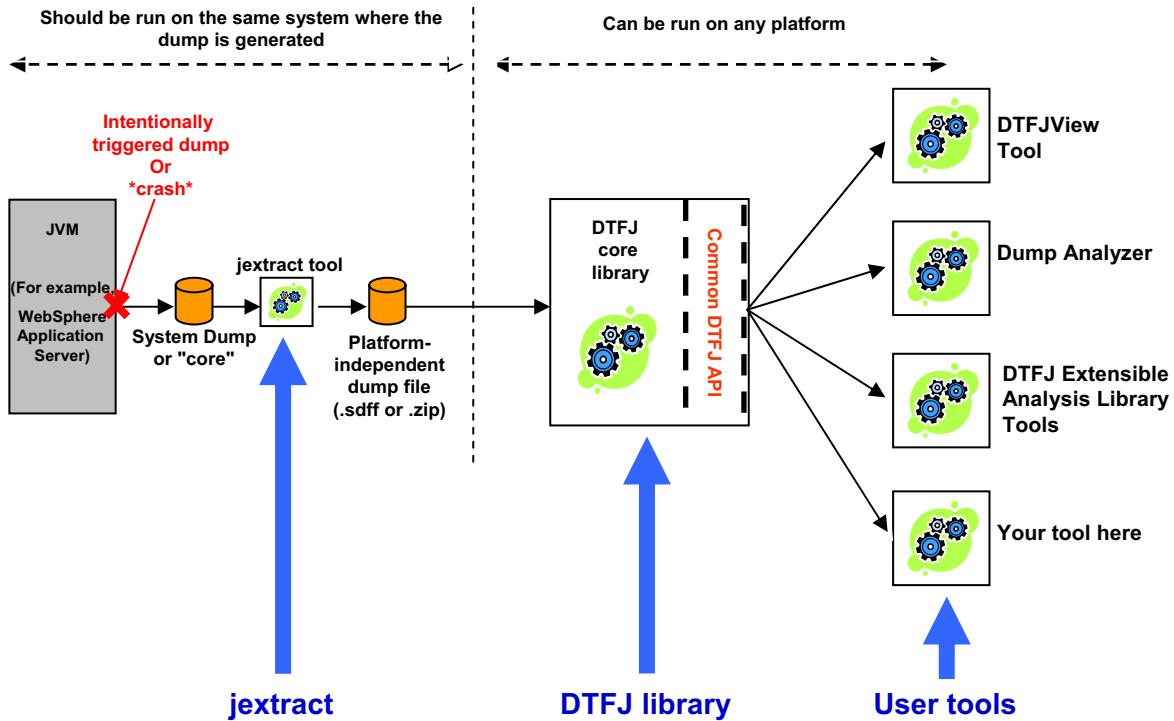


Figure 8-20. Using the DTFJ components - Example

WA5711.0

Notes:

In particular, DTFJ can examine a system core dump from a JVM and produce information readable by tools such as the Dump Analyzer.

DTFJ in this context is a set of three distinct components:

- A program called *jextract*, that processes a core file and produces a platform-independent file suitable for the main phase of analysis
- A library, the DTFJ dump reader library, that performs the bulk of the analysis
- A program called a *Dump Analyzer* that drives the library to extract particular pieces of information for processing

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Where is DTFJ supported?

- jextract + the main DTFJ runtime library are now shipped and supported with the standard IBM JDK
 - IBM JDK 1.4.2 SR4 and beyond - WebSphere Application Server 5.1, 6.0
 - IBM JDK 1.4.2 SR4 for 64-bit platforms - WebSphere Application Server 6.0.2
 - IBM JDK 1.5.0 SR1 and beyond - WebSphere Application Server 6.1
 - On all IBM JDK platforms: AIX, Linux, Windows, z/OS, iSeries
 - Including 32-bit and 64-bit

- Tools must be obtained separately

- You must be on one of these supported platforms to run the DTFJ tools
 - But you may be able to process dumps generated on an older JDK version
 - Within the same JDK family (that is, use 1.4.2 DTFJ to process any dumps from 1.4.2; use 1.5.0 DTFJ to process any dumps from 1.5.0)
 - The more recent the JDK version, the more information jextract+DTFJ will be able to extract from that dump

- Not currently supported for non-IBM JDKs such as Sun and HP

Figure 8-21. Where is DTFJ supported?

WA5711.0

Notes:

To emphasize, DTFJ supports only the IBM JDKs beginning with JDK 1.4.2 SR4. This includes IBM JDK 5 which is used by WebSphere Application Server v6.1. JDK 1.5.0 is also known as JDK 5.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using the Dump Analyzer tool

- Must be running on one of the supported JDK platforms/versions
- Distributed and accessed through IBM Support Assistant
- Based on DTFJ providing cross platform support
- Built-in analysis modules
 - Analyze the dump
 - Answer simple questions:
 - Did you run out of memory?
 - Is the JIT active?
 - Is this a WebSphere dump?

Figure 8-22. Using the Dump Analyzer tool

WA5711.0

Notes:

For information about how to enable system dumps in the IBM JVM and running jextract, see the IBM JDK Diagnostics Guide at <http://www-128.ibm.com/developerworks/java/jdk/diagnosis/>

Instructor notes:

Purpose —

Details — Also available outside of ISA but requires internal access:

Download the tool from <https://cs.opensource.ibm.com/projects/dumpanalyser/>

Unzip the package. Usage instructions will be in the README file.

Additional information —

Transition statement —

Dump Analyzer features

- Attempts to diagnose common JVM problems
 - Deadlock in Java code
 - Report thread names, locations, and so on
 - Out of memory condition
 - Report populations and large collections, and so on
 - Summarise the native memory usage
 - Recommend further analysis using MDD4J
 - Analysis generally requires multiple heap dumps
 - Internal error (gpf, and so on)
 - Is failure in non-IBM native code?
 - Probably user coding error, report location, and so on
 - If using JDK V5, it may recommend running with `-Xcheck:jni`
 - Otherwise, call IBM Support
- Otherwise, generates a default summary report
 - Recommended action is to call IBM Support and provide the output

Figure 8-23. Dump Analyzer features

WA5711.0

Notes:

If one of the common JVM problems is not found, the general script generates a default summary report.

Specifically for crashes, Dump Analyzer can read core files and attempt to determine the cause of a JVM crash. This can be extremely useful in cases where a javacore is not available. Obviously the output can be useful on other common JVM problems such as hangs and OutOfMemory conditions.

The `-Xcheckjni` command line option causes the JVM to monitor JNI usage. When `-Xcheckjni` is used, the JVM writes a warning message when more than 16 local references are required at run time. (The default root set is large enough to contain 16 local references per J2N transition) This output might indicate you should manage local references more explicitly, using the JNI functions available.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Dump Analyzer default report contents

- Basic information about the JVM process
 - Processor type, process ID, command line, JVM version, and so on
- JVM initialization arguments
 - System class path, heap tuning parameters, and so on
- Environment variables
- Native libraries loaded in this process
- Threads (both Java threads and native threads)
 - Java thread ID, WAS thread ID, java.lang.Thread object, priority, and so on
 - Java stack, native stack
- Heap memory layout
- Plus additional information

Figure 8-24. Dump Analyzer default report contents

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Dump Analyzer: Initialization

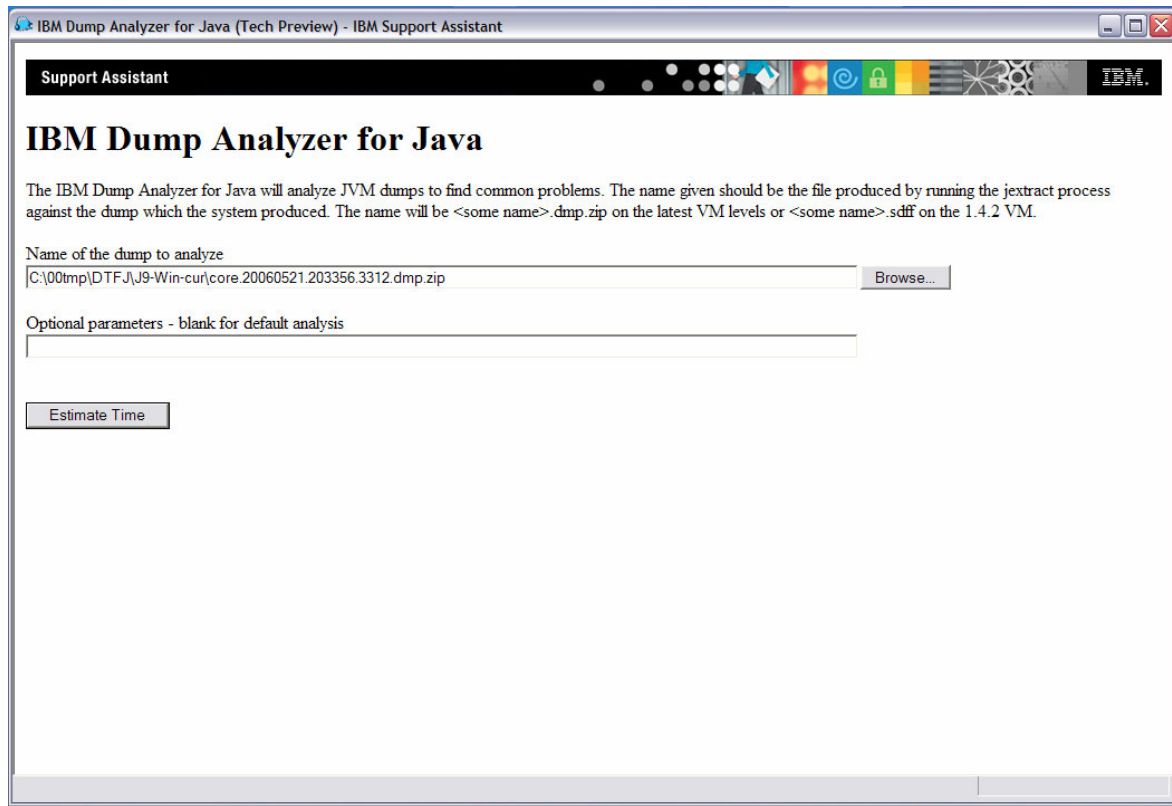


Figure 8-25. Dump Analyzer: Initialization

WA5711.0

Notes:

Browse to or input the location and name of the dump file and the tool will provide an estimate on the time to parse the file.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Dump Analyzer: Time estimated

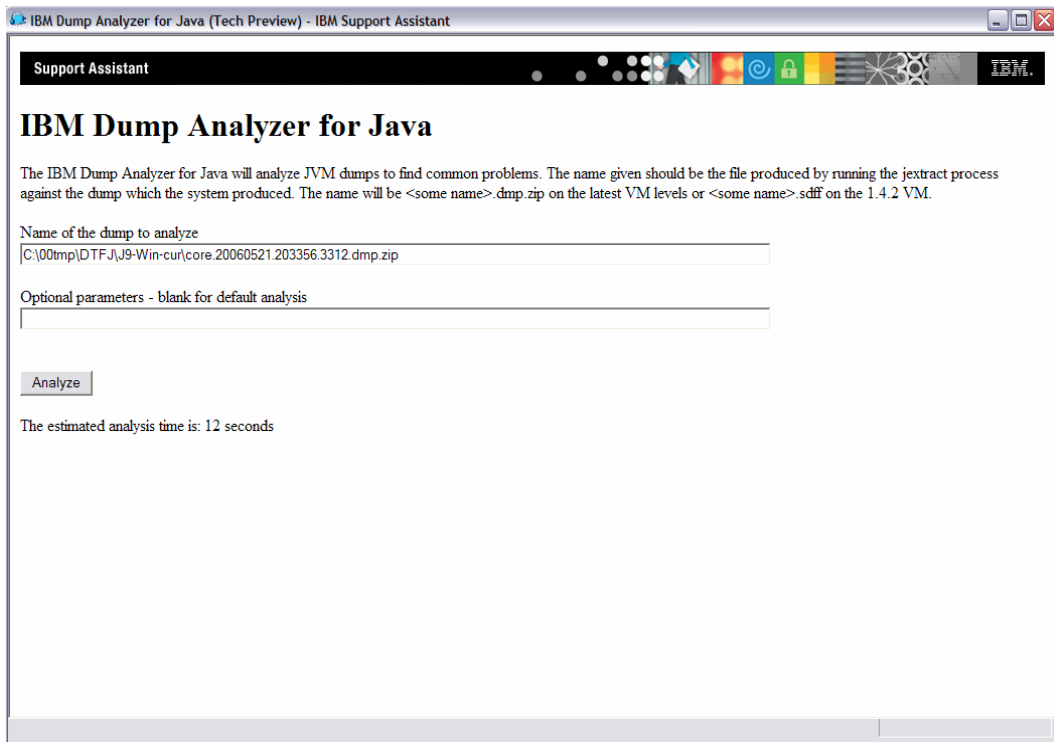


Figure 8-26. Dump Analyzer: Time estimated

WA5711.0

Notes:

An estimate of the time needed to parse the file will be provided.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Dump Analyzer: Analysis completed

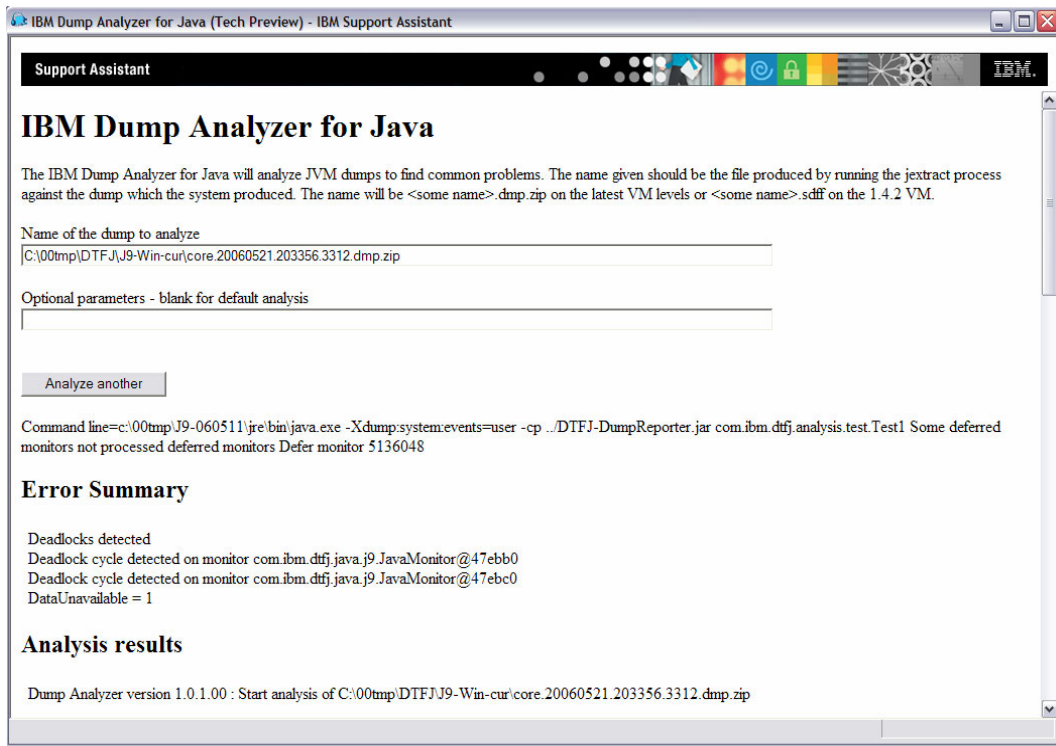


Figure 8-27. Dump Analyzer: Analysis completed

WA5711.0

Notes:

Once the analysis is completed the results will be displayed. The above example indicates that a deadlock was detected.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Dump Analyzer deadlock example detail

```

Error Summary

Deadlocks detected
Deadlock cycle detected on monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebb0
Deadlock cycle detected on monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebc0
DataUnavailable = 1

Analysis results

Dump Analyzer version 1.0.1.00 : Start analysis of C:\00tmp\DTFJ\J9-Win-
cur\core.20060521.203356.3312.dmp.zip

Start Report

Dump name: C:\00tmp\DTFJ\J9-Win-cur\core.20060521.203356.3312.dmp.zip
Creation time: Sun May 21 20:33:56 EDT 2006
System : Windows XP / 5.1 build 2600 Service Pack 2
Processor : x86 / Level 6 Model 9 Stepping 5
Installed memory: 2146353152
Processor count: 1
Command line: c:\00tmp\J9-060511\jre\bin\java.exe -Xdump:system:events=user -cp ../DTFJ-DumpReporter.jar
com.ibm.dtfj.analysis.test.Test1
Process ID: 3312
Executable: c:\00tmp\J9-060511\jre\bin\java.exe
Pointer size: 32
Current thread null
JIT is active in this run
Error Deadlocks detected
Deadlock cycle detected
0 monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebb0 is owned by thread Test1Thread1 which is waiting on ...
1 monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebc0 is owned by thread Test1Thread2 which is waiting on ...
2 monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebb0 is owned by thread Test1Thread1 which is waiting on ...
Deadlock cycle detected
0 monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebc0 is owned by thread Test1Thread2 which is waiting on ...
1 monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebb0 is owned by thread Test1Thread1 which is waiting on ...
2 monitor com.ibm.dtfj.java.j9.JavaMonitor@47ebc0 is owned by thread Test1Thread2 which is waiting on ...
Thread Test1Thread2 : owned monitors and top 2 frames on stack
owns com.ibm.dtfj.java.j9.JavaMonitor@47ebc0
frame 1 com/ibm/dtfj/analysis/test/LockingClass::syncMethod
(Ljava/lang/String;Lcom/ibm/dtfj/analysis/test/LockingClass;)V line 52
frame 2 com/ibm/dtfj/analysis/test/Test1::run ()V line 86
Thread Test1Thread1 : owned monitors and top 2 frames on stack
owns com.ibm.dtfj.java.j9.JavaMonitor@47ebb0

```

Figure 8-28. Dump Analyzer deadlock example detail

WA5711.0

Notes:

Though a javacore will point out a deadlock situation, at times the process may be so bad that a javacore cannot be generated and using a system dump is the only option.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Dump Analyzer default report example

```

=====
DTFJ DefaultDumpReport: report basic information from this JVM image (similar to javacore)
=====

Analyzer name: com.ibm.dtfj.analyzer.deal.basic.DefaultDumpReport
Analyzer version: 1.3.0.20070417
Analysis level: 1

Old-style report options (after parsing): : -sections:LI1:HP16:TH114:IA1

===== Image and runtime information =====

Now reporting on runtime 0.0.0

Image: com.ibm.dtfj.sov.imp.linux.ImageEffigy@4c464c46
Time of dump: Thu Jun 28 08:10:56 GMT+00:00 2007
System Type: Linux                System SubType: SUSE
Processor Type: Intel processor   Processor SubType: UNKNOWN PROCESSOR SUBTYPE (-1)
Number of Processors: [<unavailable>]
Installed Memory: [<unavailable>]
Host Name: [<unavailable>]
IP addresses: [<unavailable>]
This Image contains: 1 address spaces; 1 processes; 2 runtimes

Process: PID:9583
Executable: [<unavailable>]
Command line: /opt/IBM/WebSphere/AppServer/java/bin/java -
Xbootclasspath/p:/opt/IBM/WebSphere/AppServer/java/jre/lib/ext/ibmorb.jar:/opt/IBM/WebSphere/AppServer/java/jre/lib/ext/ibm
ext.jar -Dwas.status.socket=34912 -classpath
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/properties:/opt/IBM/WebSphere/AppServer/profiles:/opt/IBM/WebSphere/AppSer
ver/lib/bootstrap.jar:/opt/IBM/WebSphere/AppServer/lib/j2ee.jar:/opt/IBM/WebSphere/AppServer/lib/lmproxy.jar:/opt/IBM/WebSp
here/AppServer/lib/urlprotocols.jar -Xms1024m -Xmx2048m -
Dws.ext.dirs=/opt/IBM/WebSphere/AppServer/java/lib:/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/classes:/opt/IBM/WebSpher
e/AppServer/classes:/opt/IBM/WebSphere/AppServer/lib:/opt/IBM/WebSphere/AppServer/installedChannels:/opt/IBM/WebSphere/AppS
erver/lib/ext:/opt/IBM/WebSphere/AppServer/web/help:/opt/IBM/WebSphere/AppServer/deploytool/itp/plugins/com.ibm.etools.ejbd
eploy/runtime -Dderby.system.home=/opt/IBM/WebSphere/AppServer/derby -
Dcom.ibm.itp.location=/opt/IBM/WebSphere/AppServer/bin -Djava.util.logging.configureByServer=true -
Dibm.websphere.preload.classes=true -Duser.install.root=/opt/IBM/WebSphere/AppServer/profiles/AppSrv01 -
Dwas.install.root=/opt/IBM/WebSphere/AppServer -Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager -
Ddb2j.system.home=/opt/IBM/WebSphere/AppServer/cloudscape -

```

Figure 8-29. Dump Analyzer default report example

WA5711.0

Notes:

In this example, no common JVM errors were detected in the core file so a report similar to a javacore is created.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic summary

Having completed this topic, you should be able to:

- Describe the Diagnostic Tooling Framework for Java (DTFJ)
- Know how to process a core dump using the Dump Analyzer for Java

Figure 8-30. Topic summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe what a crash is and be able to detect one
- Analyze javacore files for crash
- Use the Dump Analyzer when javacore files are not available

Figure 8-31. Unit summary

WA5711.0

Notes:

This concludes this unit.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 3: Hung threads and server crash detection

Figure 8-32. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 9. Out-of-memory conditions

Estimated time

00:45

What this unit is about

This unit describes the how to troubleshoot out-of-memory conditions.

What you should be able to do

After completing this unit, you should be able to:

- Define an out-of-memory condition
- Use Tivoli Performance Viewer to detect out-of-memory conditions
- Obtain and interpret a verboseGC log
- Obtain and interpret a heap dump
- Discuss tools for analyzing out-of-memory problems

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

SG24-6798-00 *WAS V6 Problem Determination for Distributed Platforms*

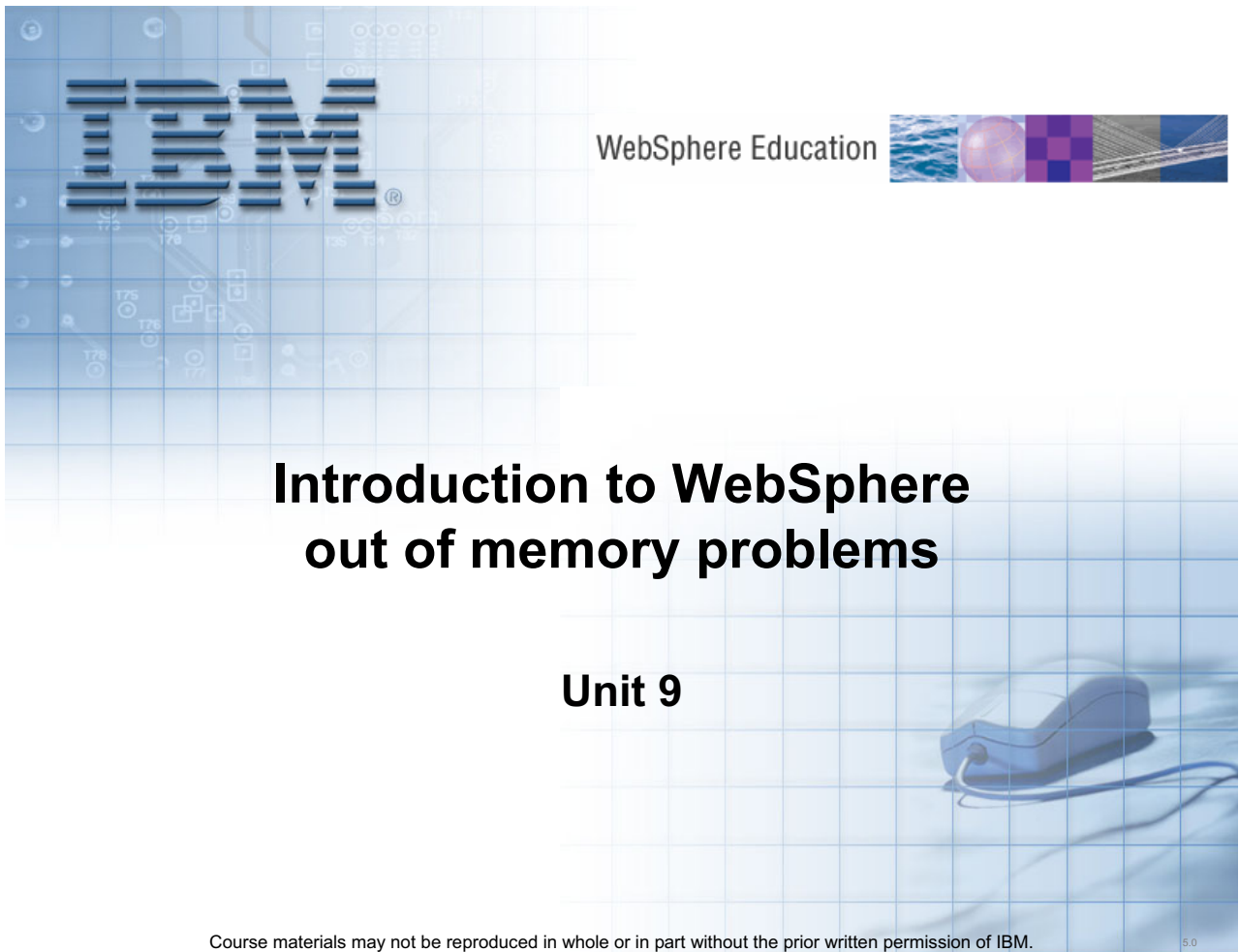


Figure 9-1. Introduction to WebSphere out of memory problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe what causes OutOfMemory conditions
- Use the Tivoli Performance Viewer and javacore files to detect OutOfMemory conditions
- Obtain a VerboseGC log
- Obtain a Java heap dump
- Interpret a verboseGC log
- Analyze a Java heap dump

Figure 9-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose — Identify and understand OutOfMemory conditions and how to collect and analyze diagnostic information required to resolve them. The unit focuses on WebSphere Application Server V6.1 and therefore JDK 5. It does cover JDK V1.4.2 for students using older versions of WebSphere. The appendix walks students through reading the verboseGC output from JDK 1.4.2 to augment the verboseGC output slides for JDK 5 including in the unit.

Details —

Additional information —

Transition statement —

OutOfMemory error overview

After completing this topic, you should be able to:

- Describe conditions that lead to memory issues
- Know how to use the Tivoli Performance Viewer monitor heap usage
- Identify javacore file OutOfMemory exception indicators
- Obtain verbose garbage collection output
- Enable the generation of heap dumps

Figure 9-3. OutOfMemory error overview

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

What is a java.lang.OutOfMemory error?

- Java virtual machine error
- Not enough memory to allocate an object; can be caused by the following:
 - The Java heap is too small
 - Memory is available in the heap, but it is fragmented (for JDK 1.4.2 and earlier)
 - Memory leak in the Java code
 - Not enough space in the native memory
- If available, first analyze the javacore file for memory issues:
 - Verify OutOfMemory exception indicated
 - Check heap size information
- Generate and analyze garbage collection and heap information:
 - VerboseGC
 - Heap dumps

Figure 9-4. What is a java.lang.OutOfMemory error?

WA5711.0

Notes:

When the Java virtual machine (JVM) tries to allocate an object and it fails, it runs garbage collection to free up heap space being used by objects that are no longer being used. If the object still cannot be allocated after garbage collection, the JVM throws `java.lang.OutOfMemoryError`.

There are four main conditions that can lead to an `OutOfMemoryError` error:

1. The maximum size of the JVM heap is too small.
2. There may be enough unallocated space in the JVM heap, but it is not contiguous. This is also called fragmentation.



Note

Fragmentation is not present in JDK V5.

3. The Java code continuously instantiates objects but some are never relinquished even when no longer used. This is called a memory leak.
4. The final condition is due to insufficient space in the native memory segment. Perhaps there is insufficient native memory but this can also be the result of a memory leak in the native memory segment.

These conditions will be described on the following slides.

If an `OutOfMemory` exception was encountered it should be indicated in the `SystemOut.log` file. Then look to see if a `javacore` file was created. If so—use the file to get an initial idea of the nature of the `OutOfMemory` condition. Then investigate GC history and heap data for more detailed information. Heap data is required to help identify memory leaks.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JVM heap is too small

- How do you know the heap is too small?
 - The JVM heap usually displays constant growth until it reaches the maximum heap size
 - The JVM heap never achieves a steady state
- Need to increase the Java maximum heap size in the administrative console
 - **Servers -> Application Servers -> server -> Java and Process Management -> Process Definition -> Java Virtual Machine -> Maximum heap size**
 - Set as part of a performance tuning process outlined in the information center
 - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/tprf_tunejvm_v61.html

Figure 9-5. JVM heap is too small

WA5711.0

Notes:

Applications require a minimum amount of memory to reach a stable state. The stable state occurs when the heap is no longer consistently growing. To reach a stable state, the application has to run through its commonly used code paths and instantiate all of the frequently referenced objects. The load used to test for the stable state is very important. Usually more memory is required to reach a stable state under higher loads. If the JVM is configured with a maximum heap that is too small and never allows the JVM to reach a stable state, then allocation failures will occur and the JVM will throw `OutOfMemoryError`.

Applications should go through rigorous performance tuning before being deployed into a production environment.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Memory fragmentation (JDK 1.4.2 and earlier)

- The difference between the maximum heap size and the current heap size is only slightly greater than the size of the object to be allocated.
 - Some fragmentation will always occur.
 - Should be treated as if the heap was too small.
- The object being allocated is excessively large.
 - The JVM is attempting to allocate an object that takes up a significant portion of the heap by itself.
 - The application developer should attempt to reduce the size of the object being allocated. If that is not possible, increase the JVM heap.

Figure 9-6. Memory fragmentation (JDK 1.4.2 and earlier)

WA5711.0

Notes:

In order to avoid fragmentation, the JVM 5.0 may end up doing a lot of compaction, which is CPU-intensive. So if your program allocates lots of large objects and causes a lot of fragmentation, you may end up seeing horrible performance, even if you do not crash with OOM.

When memory is allocated for an object, the object gets a range of contiguous memory from the heap. The next object will get another range of contiguous memory addresses from the heap. At the same time memory is being allocated, it is being deallocated and returned to the heap. Because memory is constantly in the process of being allocated and deallocated, it is impossible for all available memory in the heap to be in one contiguous block. Instead, chunks of allocated memory are fragmented throughout the heap. To be able to allocate an object, memory must be in a contiguous block. It is possible that there is enough available memory in the heap to fulfill an allocation request, but there is not a single block of contiguous memory available so the request fails.

There are two specific cases of fragmentation that are commonly seen. The first occurs when the difference between the current heap size and the maximum heap size is only

slightly greater than the size of the object to be allocated. Since some fragmentation will always occur within any memory management system, this case should be treated as if the heap size was too small.

The second case is when the JVM continuously attempts to allocate excessively large objects. To illustrate the problem of large object allocations, imagine the Java heap as a stack of seven blocks. Each block represents one unit of memory. Now imagine that you have an allocation request for three units of memory. If the memory could be allocated from the top or bottom of the heap, there would still be four contiguous memory units available. You would still be able to satisfy an allocation request for four memory units. Now imagine that the only contiguous three blocks available are the center three, so you allocate them. Soon after, all other blocks become deallocated, thus there are four available units of memory. Unfortunately, since the three units in the middle of the heap are allocated, that leaves only two contiguous units on each end. This means that any request to allocate an object that requires more than two units of memory will fail. Fragmentation becomes more of a problem as the allocation requests get larger.

In the second case, the best practice is to reduce the size of the object being allocated. If this is not possible, then the only other alternative is to increase the size of the JVM heap.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Memory leak in the Java code

- No matter what the JVM maximum heap size is set to, the heap will still run out of space.
- Increasing the maximum heap size only causes the problem to take longer to occur.
- One or more objects are taking up a high percentage of the JVM heap:
 - A few large objects
 - Thousands of instances of small objects
- Memory leaks can also occur in native code

Figure 9-7. Memory leak in the Java code

WA5711.0

Notes:

Though memory is not explicitly allocated and deallocated in Java, it is still possible to create a memory leak. One example would be to save an object into some type of collection. If the collection is a class object, and the class always stays loaded, the object will never be removed from the collection. If objects are continuously added to the collection, the collection could grow until it consumes a significant portion of the Java heap.

The misuse of object caches is a common cause of memory leaks seen in applications running in WebSphere. One example is the configuration of a large PreparedStatement cache. The PreparedStatement cache size is the total number of PreparedStatements that WebSphere will maintain in the cache. PreparedStatement objects are the parent objects of ResultSet objects, where data from a PreparedStatement is stored. If the ResultSet objects associated with the PreparedStatements in cache are fairly large, and the PreparedStatement cache size is large, then the PreparedStatement cache can grow very quickly and consume significant amounts of available memory.

Another commonly seen misuse of cache in WebSphere is the HTTP max Sessions parameter in WebSphere. HTTP Sessions can be written to store very large objects. If the

HTTP max sessions parameter is set too high, then it will again consume a large portion of the JVM heap.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Not enough native memory

- Insufficient memory available in the native memory segment
- There is more than sufficient space in the JVM heap, but the allocation still fails
- The JVM is not necessarily trying to allocate a large object but rather memory is not available in the native memory space

Figure 9-8. Not enough native memory

WA5711.0

Notes:

When the Java virtual machine is unable to find enough contiguous memory in the heap for object allocation, and it has already called the garbage collection routine, it will then attempt to grow the JVM heap up to the maximum heap size. In order to cause the JVM heap to grow, the JVM must request, and be granted, system memory from the operating system Memory Management Unit (MMU). If the MMU is unable to provide memory to the JVM for heap expansion, the JVM will throw `OutOfMemoryError`.

Every Java thread running within a JVM also has an associated thread stack, which requires system memory. If there is not enough memory in the system for the allocation of a Java thread, the system will again throw `OutOfMemoryError`.

Finally, through the Java Native Interface (JNI) the application can access system libraries, which require native system memory to be loaded. If the application fails to remove unused references to JNI objects thus causing a memory leak, or if the system is low on native memory to begin with and is unable to provide memory for the object allocation, the JVM will throw `OutOfMemoryError`.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Java process restrictions

- Not all Java process space is available to the Java application heap
- The Java run time needs memory for:
 - The Java virtual machine resources
 - Backing resources for some Java objects

- This memory area is part of the native heap
- Memory not allocated to the Java heap is available to the native heap
 - Available memory space – Java heap = native heap

- Effectively, the Java process maintains two memory pools:
Java and native

Figure 9-9. Java process restrictions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

The native heap

- Allocated using **malloc()** and therefore subject to memory management by the OS
- Used for virtual machine resources:
 - Execution engine
 - Class Loader
 - Garbage Collector infrastructure
- Used to underpin Java objects:
 - Threads, Classes, AWT objects
- Used for allocations by JNI code
- Size can not directly be controlled. To increase this memory space decrease the Java heap size.

Figure 9-10. The native heap

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using TPV to anticipate an OutOfMemory error

- Tivoli Performance Viewer (TPV) runs in the WebSphere Administrative Console
- Uses Performance Monitoring Infrastructure (PMI) to capture information about the WebSphere run time.
- Provides graphical display of the Captured PMI Data.

Figure 9-11. Using TPV to anticipate an OutOfMemory error

WA5711.0

Notes:

The Tivoli Performance Viewer is embedded within the WebSphere Administrative Client. It uses the PMI infrastructure to capture information about the WebSphere run time, such as the JVM heap size. You can change the PMI settings in the WebSphere console to capture the information you want to see, set the TPV to begin logging the PMI request information, and view the graphical representation of the data collected. In this case, you want to monitor the size of the JVM heap.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative Console PMI settings

- Select **Performance and Tuning -> PMI -> server_name**

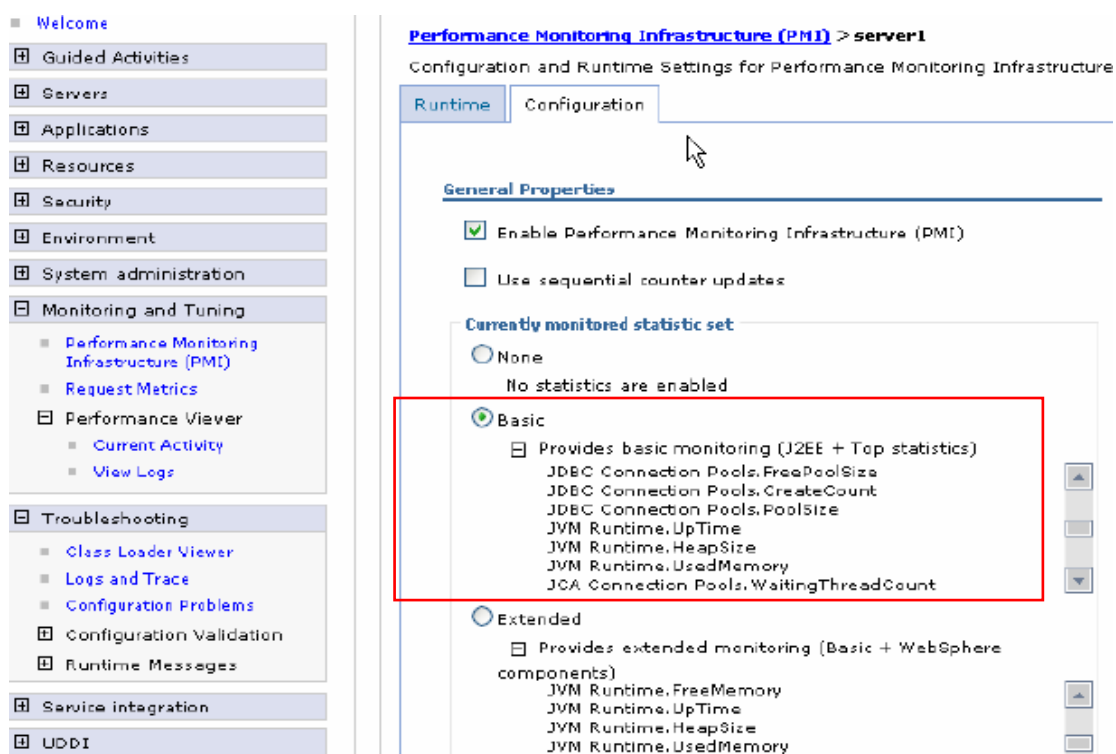


Figure 9-12. Administrative Console PMI settings

WA5711.0

Notes:

Under **Monitoring and Tuning**, select the **Performance Monitoring Infrastructure (PMI)** link. Under the **Configuration** tab, select the check box to enable PMI. Under **Currently monitored statistic set** select **Basic**. From the picture you see this includes **JVM Runtime.HeapSize** statistics.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative Console TPV monitoring

- Select **Monitoring and Tuning** -> **Performance Viewer** -> **Current Activity** -> **server_name**
- Click **Start Monitoring** button

The screenshot shows the Tivoli Performance Viewer interface. On the left is a navigation tree with 'Monitoring and Tuning' expanded to 'Performance Viewer' and 'Current Activity' selected. The main panel shows a message box stating 'Monitoring has started for server server1 on node hgloverNode01.' Below this, there are 'Start Monitoring' and 'Stop Monitoring' buttons, with 'Start Monitoring' circled in red. A table below shows the server details:

Select	Server	Node	Version	Collection Status
<input type="checkbox"/>	server1	hgloverNode01	6.0.2.0	Monitored
Total 1				

Figure 9-13. Administrative Console TPV monitoring

WA5711.0

Notes:

Under **Performance Viewer**, select the **Current Activity** link. Select the application server you wish to monitor, and click the **Start Monitoring** button. Then select the link to your application server, in this case **server1**, and hit the **Start Logging** button.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative Console TPV Graph

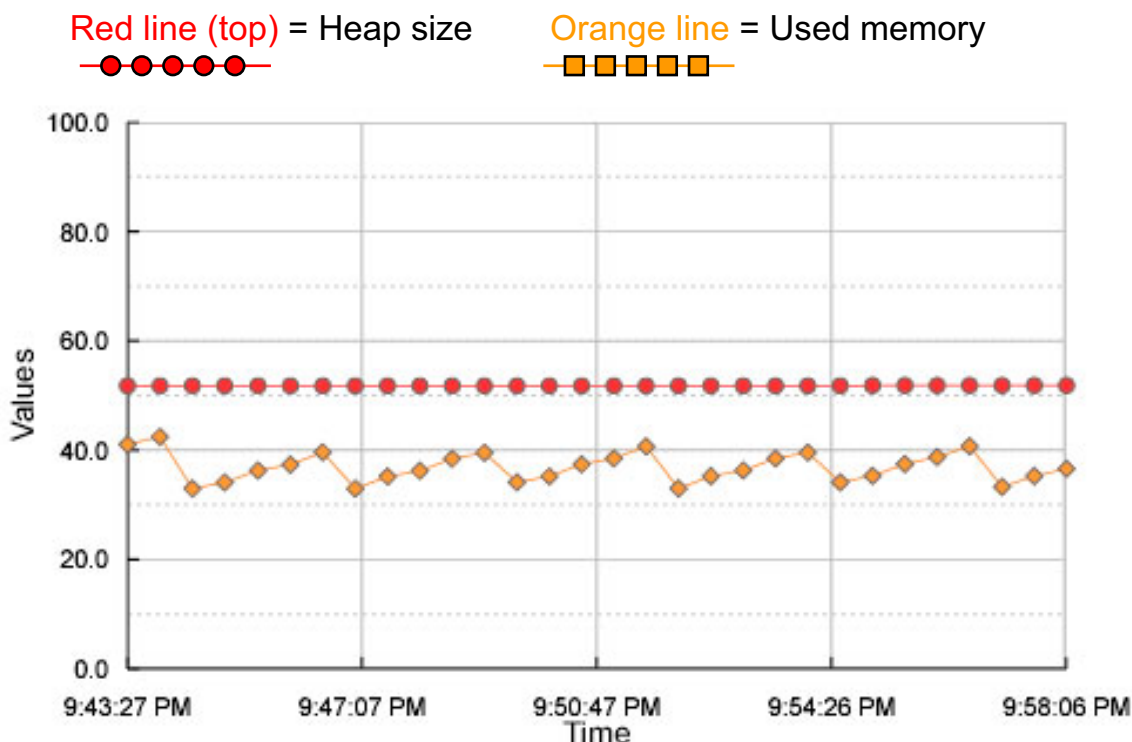


Figure 9-14. Administrative Console TPV Graph

WA5711.0

Notes:

You may need to leave that page and come back in, but the TPV will provide a graph that includes the JVM heap size of the WebSphere run time over time. In this case the heap size is the red line, and the free memory is represented by the orange line. You can monitor these values to determine when an OutOfMemory condition is about to occur.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

TPV usage

- Tivoli Performance Viewer (TPV)
 - Monitor internal thread pools and heap statistics
- Know the expected behavior of your application – I/O intensive, JMS, number of EJBs, expected load requirements, and so on
- Use TPV to monitor the heap usage
 - If the used heap continues to grow overtime with no corresponding increase in user load there may be a memory leak
 - Use a profiler to dig into the specific heap to determine where the problem may exist

Figure 9-15. TPV usage

WA5711.0

Notes:

The Tivoli Performance Viewer (TPV) can be used to monitor overall heap usage and the internal threads. If needed, even more specific information can be collected on garbage collections, but this option adds a large overhead to the process. See the information center for details.

TPV is an integrated tool provided with WebSphere. There are more sophisticated monitoring tools available from Tivoli and third party vendors.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

OutOfMemory indicators of the javacore file

- Look to see if the dump was due to an OutOfMemory condition

```
1TISIGINFO      Dump Event "throw" (00000010) Detail
  "java/lang/OutOfMemoryError" received
```

- Check MEMINFO subcomponent output (values in hexadecimal)

```
0SECTION      MEMINFO subcomponent dump routine
NULL          =====
1STHEAPFREE   Bytes of Heap Space Free: 4bca320
1STHEAPALLOC  Bytes of Heap Space Allocated: c000000
```

- Check GC History

```
1STGCHTYPE    GC History
3STHSTTYPE    05:15:28:807904721 GMT j9mm.81 -
  J9AllocateIndexableObject() returning NULL! 167772176 bytes
  requested for object of class 101189C8
```

- Free memory indicator

- If the free space is tight, increasing the size of the heap may solve the problem
- If there is a lot of free space available, the problem may be due to a large object allocation or problem with native memory allocation

Figure 9-16. OutOfMemory indicators of the javacore file

WA5711.0

Notes:

Since the javacore file provides the command line arguments for the JVM, you can see the configured maximum heap size using `-Xmx` argument if provided. If not present then the default maximum is used. You can also see the `-Xdump` dump agent settings configured for the JVM.

The MEMINFO and GC History output is new to JDK 5.

If an OutOfMemory exception was encountered it should be indicated in the SystemOut.log file. Then look to see if a javacore file was created. If so, use the file to get an initial idea of the nature of the OutOfMemory condition. Then if needed perform a more detailed analysis investigating heap data and GC history.

If free space is tight, try increasing the maximum heap size. Be sure to take into account possible memory leaks. If over time the heap is being consumed without a corresponding increase in user load it is probable that there is a memory leak.

Check the GC History output. In the example above you can see that an allocation attempt was made for approximately 167M. The MEMINFO component shows that there was only

around 79M (hexadecimal 4bca320) free. In this example it is evident that an allocation for a large object led to the OutOfMemory condition.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

How to obtain a verboseGC log

- Verbose GC is an option provided by the JVM run time
- Enables a garbage collection log
 - Interval between collections
 - Duration of collection
 - Compaction required
 - Memory size/memory freed/memory available
- Enable the verbose GC for the server using the administration console
 - Navigate to **Servers -> Application servers -> server_name**
 - Under Server Infrastructure, click **Java and Process Management -> Process Definition -> Java Virtual Machine**
 - Select **Verbose Garbage Collection** check box
 - Save and distribute
 - Restart the server or servers
- Usually writes to native_stderr
 - Varies depending on platform and WebSphere version
 - Some overhead because of disk I/O, but usually minimal unless thrashing

Figure 9-17. How to obtain a verboseGC log

WA5711.0

Notes:

In version 6.1, it is possible to enable verbose GC output using the **Runtime** tab. This allows the initiation of verbose GC output on a running application server.

It is often recommended to have verbose GC enabled permanently in production. The overhead cost on a reasonably well-tuned JVM is actually quite small. The benefits of having it on the first time something happens are considerable (no need to reproduce the problem a second time after enabling). It is also good to keep an eye on the verbose GC regularly simply as a way to monitor the health of the system, even when nothing bad has been noticed.

So of course enabling verbose GC by default is a decision that must be made consciously by each system administrator. But it is no longer plainly “not recommended as a normal production setting.”

The verboseGC (verbose garbage collection) output is the diagnostic data used to identify what type of OutOfMemoryError condition is occurring on the system. The verboseGC output shows the size of the object that is being allocated, how much memory is available on the Java heap, and what the response of the JVM is to the allocation request. In many

cases the VerboseGC output will give you a good idea how to resolve the problem, and even if it doesn't it will help you to narrow the problem.

Use the following to initiate verbose GC on non-IBM JDKs:

- For HP-UX, add the following parameter to the Generic JVM Arguments on the Java virtual machine settings page:

```
-Xverbosegc:file=<name>
```

- For Solaris, add the following parameter to the Generic JVM Arguments on the Java virtual machine settings page:

```
-XX:+PrintGCDetails -XX:+PrintGCtimeStamps -XX:+PrintHeapAtGC
```

There are some small differences in the VerboseGC output between the Sun, HP-UX, and IBM JVMs. The IBM JVM (Windows, AIX, and Linux) produces highly detailed output by default and places the information in the `native_stderr.log` file. The HP-UX JVM also provides detailed output, but you have to tell it where to write the output. The Sun JVM on Solaris will write the verboseGC output to the `native_stdout.log` file, but it does not provide detailed output by default. You have to provide the parameters to tell it to print out the detailed information about the GC event, the timestamp of the GC event, and the size of the JVM heap at the time of the GC event.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Obtaining Java heap dumps

- Used for analyzing the actual contents of the Java heap
 - Shows the objects using the heap memory
 - Needed for memory leak debugging
 - Heap dumps can be created by default for OutOfMemory exceptions
- Generated from a running JVM in these ways:
 - Explicit/manual generation – such as using wsadmin
 - JVM-triggered generation – using dump agents to handle exceptions
 - Automated heap dump generation facility
- IBM Java heap dump for IBM JVMs
 - Windows, AIX and Linux
 - JDK v5.0 and later – use command line argument `-Xdump:heap`
 - JDK v.1.4.2 and earlier – use environment variables
- Java memory dumps on Solaris
 - IBM Heapdump Agent for 1.4.2_08 and earlier
 - JMAP for 1.4.2_09 – 1.4.2_11
 - Sun heapdump functionality for 1.4.2_12 and above
- Java memory dumps on HP-UX
 - HPROF

Figure 9-18. Obtaining Java heap dumps

WA5711.0

Notes:

A memory dump of the JVM heap is the primary diagnostic data for identifying memory leak culprits. With the use of special tools, you can analyze the heap dumps to tell you what objects are taking up significant chunks of heap memory. You can also tell who is responsible for instantiating those objects.

For IBM JVMs, which run on Windows, AIX, and Linux platforms, you obtain IBM Java heap dumps. The JVM is easily configured within the administrative console to generate an IBM heap dump. The advantage of the IBM heap dump is that there are tools available to parse and process the heap dump file.

On Solaris, the tool depends on the version of the Sun JVM. Depending on the version of the JVM you will use the IBM Heapdump Agent, JMAP, or Sun's own heap dump functionality.

For HP-UX, the suggested tool for obtaining a dump of the Java heap is the Heapdump agent.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Use wsadmin to trigger a heap dump

- From a command prompt start the wsadmin shell by typing:
`wsadmin`
- Execute the following wsadmin commands:

```
wsadmin> set objectName [$AdminControl queryNames  
    WebSphere:type=JVM,process=<servername>,node=<noden  
    ame>,*]  
<wsadmin> $AdminControl invoke $objectName  
    generateHeapDump
```

Figure 9-19. Use wsadmin to trigger a heap dump

WA5711.0

Notes:

The default location for the placement of the javacore file is:

`<WAS_install_root>/profiles/<profile>`. See the JVM introduction unit for complete details.

A javacore file can be generated on demand through the wsadmin command line interface:

1. From the command prompt, enter the command `wsadmin.bat` to get a wsadmin command prompt.



Note

If security is enabled or the default SOAP ports have been changed, you will need to pass additional parameters to the batch file in order to get a wsadmin prompt. For example:

```
wsadmin.bat [-host host_name] [-port port_number] [-user userid[-password  
password]
```

2. Get a handle to the problem application server.

```
wsadmin> set objectName [$AdminControl queryNames  
WebSphere:type=JVM,process=<servername>,node=<nodename>,*]
```

Where *servername* is the name of the application server targeted for the heap dump. If wsadmin is connected to a dmgr and if the server names in cell is not unique, the you can qualify the JVM with node attribute in addition to process.

3. Generate the heap dump:

```
wsadmin>$AdminControl invoke $jvm dumpThreads
```

The default format for a heap dump is.phd (portable heap dump) as opposed to text format.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Java heap dumps with the IBM JDK V5

- To obtain a Java heap dump on a user signal or if the application catches the resulting signal add `-Xdump` command line options
 - From the Administrative Console select **Servers -> Application Servers -> server_name -> Java and Process Management -> Process Definition -> Java Virtual Machine**
 - Modify the **Generic JVM arguments** adding `-Xdump` command line options
 - For example, to create heapdumps even when `OutOfMemory` is caught by the application
 - `-Xdump:heap:events=throw,filter=java/lang/OutOfMemoryError`
 - Apply and save changes to the master configuration
 - Restart the application server
- Xdump examples
 - `-Xdump:heap:events=...` – enables heap dump creation for specified events
 - `-Xdump:heap:none` disable heap dump creation
 - `-Xdump:heap,opts=PHD+CLASSIC` – enables heap dump creation and creates the file in both binary and text format
- For detailed description of `-Xdump` usage:
 - <http://www.ibm.com/support/docview.wss?uid=swg21242497>

Figure 9-20. Java heap dumps with the IBM JDK V5

WA5711.0

Notes:

By default, a heap dump is generated when an `OutOfMemory` condition occurs. Using the default dump agents provided, when an `OutOfMemory` condition is encountered and the resulting exception is not caught or handled by the application, JVM-triggered generation of a heap dump occurs.

To display on JVM startup the conditions (if any) that will generate a heap dump (or `javadump` or `systemdump`), you can use `-Xdump:what`. Similar to verbose GC, the output is placed in `native_stderr.log`.

The generated heap dump is by default in the binary and platform-independent (`phd`) format which can be examined using the available tooling.

Sometimes useful to have an immediately readable view of the heap. You can obtain this view by using the `opts=` stanza with `-Xdump:heap` or by the existence of an environment variable:

IBM_JAVA_HEAPDUMP_TEST, which allows you to perform the equivalent of `opts=PHD+CLASSIC`

IBM_JAVA_HEAPDUMP_TEXT, which allows the equivalent of **opts=CLASSIC**

The JVM checks each of the following locations for existence and write-permission, then stores the heap dump in the first one that is available.

- The location that is specified using the file suboption on the triggered **-Xdump:heap** agent.
- The location that is specified by the **IBM_HEAPDUMPDIR** environment variable, if set. (**_CEE_DMPTARG** on z/OS)
- The current working directory of the JVM processes.
- The location that is specified by the **TMPDIR** environment variable, if set.
- The **/tmp** directory. On Windows, **C:\temp**.

You can filter class events (such as load, throw, and uncaught) by class name:

-Xdump:java:events=throw,filter=java/lang/OutOfMem* # prefix and wildcard

-Xdump:java:events=throw,filter=*MemoryError # suffix and wildcard

-Xdump:java:events=throw,filter=*Memory* # substring wildcards

The above examples will create a dump agents to create javacore file when an OutOfMemory exception is thrown. To specific the above to force a heap dump change **java** to **heap** (or to **java+heap**)

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Automatic heap dump generation with the IBM JDK V5

- Only works in with IBM JVMs
- Works in conjunction with the lightweight memory leak detection mechanism
- Heap dumps are generated when a memory leak is detected or when `OutOfMemoryError` is detected
- Perform the following steps in the administrative console:
 1. Click **Servers** -> **Application servers** in the administrative console navigation tree.
 2. Click **server_name** -> **Performance and Diagnostic Advisor Configuration**.
 3. Click the **Runtime** tab.
 4. Select the **Enable automatic heap dump collection** check box.
 5. Click **OK**.

Figure 9-21. Automatic heap dump generation with the IBM JDK V5

WA5711.0

Notes:

WebSphere Application Server (as of v6.02) has implemented a lightweight memory leak detection mechanism that runs within the WebSphere Runtime Performance Advisor framework. This mechanism is designed to provide early detection of memory problems in test and production environments. This framework is not designed to provide analysis of the source of the problem, but rather to provide notification and help generating the information that is required to use analysis tools. The mechanism only detects memory leaks in the Java heap and does not detect native leaks.

An automated heap dump generation facility has been provided for WebSphere Application Server running on IBM JDKs. Upon detection of a memory leak pattern, this facility will generate multiple heap dumps that have been coordinated with sufficient memory leakage to facilitate comparative analysis using MDD4J. In addition, the IBM JDK is configured to generate a heap dump automatically if an `OutOfMemoryError` is detected.

To preserve disk space, the Performance and Diagnostic Advisor does not take heap dumps if more than 10 heap dumps already exist in the WebSphere Application Server home directory. Depending on the size of the heap and the workload on the application

server, taking a heap dump might be quite expensive and might temporarily affect system performance.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Java heap dumps with the IBM JDK V1.4.2

- To obtain a Java heap dump on a user signal or if the application catches the resulting signal:
 1. From the Administrative Console select **Servers -> Application Servers -> server_name -> Java and Process Management -> Process Definition -> Environment Entries -> New**
 2. Add the following name-value pairs:

• IBM_HEAPDUMP	true
• IBM_HEAP_DUMP	true
• IBM_HEAPDUMPDIR	<i>your_dump_directory</i>
• IBM_HEAPDUMP_OUTOFMEMORY	true
• IBM_JAVADUMP_OUTOFMEMORY	true
• IBM_JAVA_HEAPDUMP_TEXT	true
 3. Apply and save changes to the master configuration.
 4. Restart the application server

Figure 9-22. Java heap dumps with the IBM JDK V1.4.2

WA5711.0

Notes:

By default, a heap dump is generated when an OutOfMemory condition occurs and resulting exception is not caught or handled by the application.

The use of the environment variable does still work in JDK V5 although the use is being deprecated. If unable to get **-Xdump** to work then try this approach.

For the IBM JVM, command line parameters must be passed to the Java executable on startup in order to obtain an IBM Java heap dump. The command line parameters can be set within the WebSphere Administrative console. After setting the parameters, the application server must be restarted in order for the JVM to be initialized with the heap dump settings.

Once set up, the IBM JVM will automatically dump a Java heap dump when an OutOfMemoryError occurs.

- **IBM_HEAPDUMP** - Enables heap dumps by means of signals
- **IBM_HEAP_DUMP** - Enables heap dumps by means of signals

- **IBM_HEAPDUMPDIR** - Path name of directory for storing heap dump files (these files can be quite large)
- **IBM_HEAPDUMP_OUTOFMEMORY** - Generates a heap dump when an out-of-memory exception is thrown. Set to false to disable heap dumps for an OutOfMemory condition
- **IBM_JAVADUMP_OUTOFMEMORY** - Trigger a Javadump on heap exhaustion/OutOfMemory error
- **IBM_JAVA_HEAPDUMP_TEXT** - Setting this variable will enable the heap dump to be generated in.txt format. The default is.phd format

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

How to obtain Java memory dumps on Solaris

- For now, contact IBM Customer Support for instructions
 - Sun JVM version 1.4.2_08 and earlier use IBM Heapagent
 - Not supported for production environments without assistance from IBM Support.
 - Sun JVM version 1.4.2_09, 1.4.2_10, and 1.4.2_11 use JMapp
 - Command line tool that ships with the Sun JVM on Solaris
 - `jmap -histo <PID>` or `jmap -histo <core file>`
 - Lists each class in the heap, the number of instances of that class, and total memory used by all instances.
 - Use the output operator to send output to a file.
 - Refer to the Sun J2SE 5.0 “Trouble-Shooting and Diagnostic Guide” for more information.
 - Sun JVM version 1.4.2_12 and higher provides ability to dump Java Heap
 - JVM command line option `-XX:+HeapDumpOnCtrlBreak`
 - SIGQUIT signal (kill -3) triggers the JVM to generate a heapdump
 - Heapdump can be parsed using MDD4J

Figure 9-23. How to obtain Java memory dumps on Solaris

WA5711.0

Notes:

The Sun 1.4.2 JVM has recently undergone a series of changes to its serviceability code. Because of that, depending on the version of the service release of the 1.4.2 JVM, there are three different options for obtaining a dump of the JVM heap.

For Sun JVM version 1.4.2_08 and earlier, IBM provides the IBM Heapdump agent which can be used to obtain an IBM Java heap dump. The advantage of the IBM Java heap dump is there are tools available to parse and analyze the file. The Heapdump agent is not supported for production environments without assistance from IBM, so users should contact IBM Support in order to obtain a heap dump for Sun JVM 1.4.2_08 and earlier.

Changes in the JVM code for Sun JVM version 1.4.2_09 will cause the JVM to crash if you attempt to attach the IBM Heapdump agent. Because of this, the IBM Java development teams suggests that you use JMapp to obtain a dump of the Java heap instead. JMapp is a command line tool that is packaged along with the Sun JVM on Solaris. JMapp can be run on a running process or a core file to obtain a dump of the Java heap.

Beginning with Sun JVM 1.4.2_12, Sun provides its own heap dump mechanism. To configure the Sun Java heap dump, set the JVM command line option, restart the

WebSphere Application Server, and then issue a **kill -3** against the WebSphere Application Server JVM process to generate a heap dump. The advantage of the Sun Java heap dump is that you will be able to parse and analyze it using MDD4J.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

How to obtain a heap dump on HP-UX

- Contact IBM Support
 - Normally use HPROF
 - Not useful with larger heap sizes which are typically used with WebSphere
 - Costly performance impact
 - Work with IBM Support find the best solution for your environment

Figure 9-24. How to obtain a heap dump on HP-UX

WA5711.0

Notes:

Obtaining a dump of the Java heap on HP-UX is more difficult than with the IBM JVMs or Solaris. HPROF is the best tool available, but it often hangs with Java heaps greater than 500 MB. Since most WebSphere Application Server JVM heap sizes are greater than 500 MB, HPROF is rarely useful. It also incurs a heavy performance cost, which negatively affects the behavior of the application. One advantage for HPROF is that there is a tool, HAT, available to parse and analyze HPROF dumps.

Users that need to obtain a dump of the Java heap for WebSphere running on HP-UX should contact IBM Support.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Interpret VerboseGC output

After completing this topic, you should be able to:

- Manually interpret verboseGC output
- Use tools to help process verboseGC output

Figure 9-25. Interpret VerboseGC output

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

OutOfMemory: Interpret verboseGC output

- Inspect the allocation failure
 - Find the allocation failure that caused the OutOfMemoryError
 - Check the size of the object to be allocated
 - Determine the size of the Java heap
 - Check to see what percentage of the heap is free

- Look for the pattern in previous allocation failures
 - Do you continuously get allocation failures that cause the JVM to increase the heap?
 - Is this an isolated allocation failure caused by a request for a large object?

Figure 9-26. OutOfMemory: Interpret verboseGC output

WA5711.0

Notes:

Once you locate the OutOfMemoryError in the log, you want to first examine the allocation failure (AF) that caused the event to occur. You need to check the size of the object to be allocated, and the current size of the Java heap. You can also observe what percentage of the heap is currently free. This allows you to determine whether garbage collection should have been able to free enough memory for the object allocation, and if not, why. For instance, it would be easy to determine that the maximum heap size was not large enough for all of the requests, or if the object to be allocated is too big relative to the JVM heap.

Once you come to a conclusion on the final AF, you should begin looking at AF events leading up to the OutOfMemoryError. Does the JVM heap keep increasing until you finally run into the Max Heap Size? Or does the AF that leads to the OutOfMemoryError seem to be an isolated event? By asking these questions you can determine which of the four causes of OutOfMemoryErrors you are experiencing.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

VerboseGC output for optthruput policy

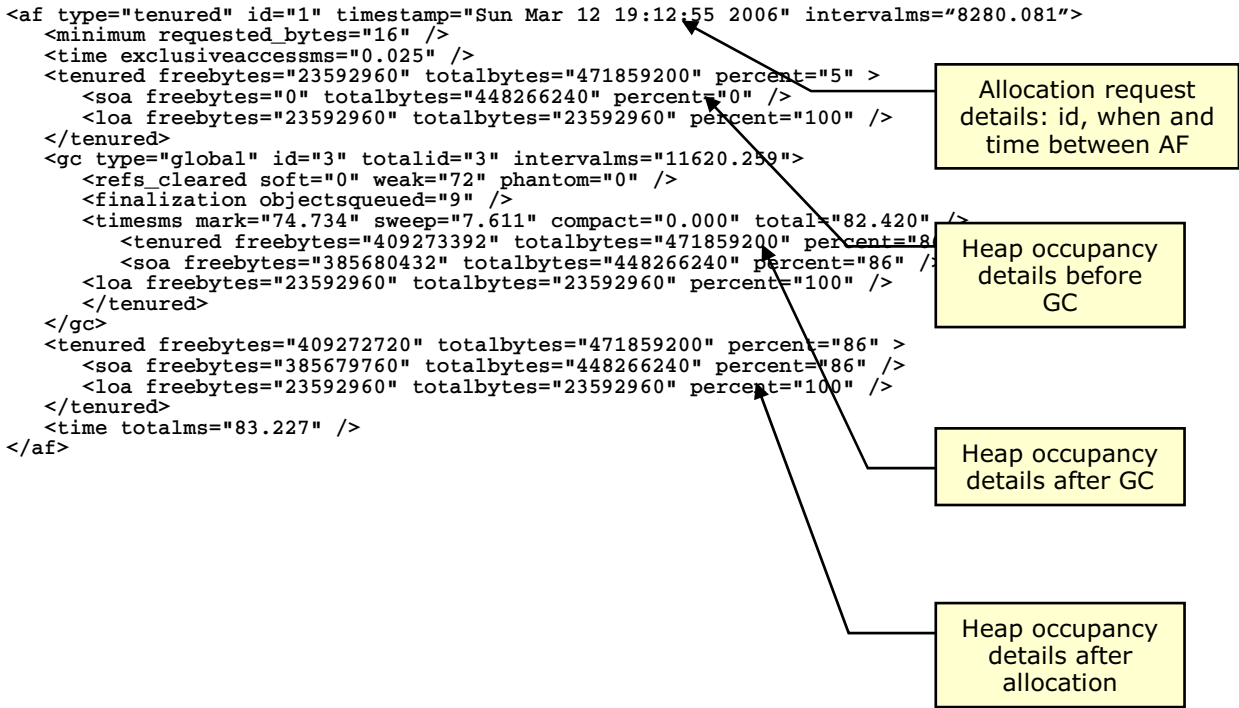


Figure 9-27. VerboseGC output for optthruput policy

WA5711.0

Notes:

The example above illustrates a normal allocation failure when using the optthruput policy. This is the default policy which disables concurrent mark. If you do not have pause time problems (as seen by erratic application response times), you get the best throughput with this option.

At the end of GC cycle there space available to allocate the requested bytes of 16. The **id** field indicates that this is the first allocation failure and **type** shows this to be a tenured collection as opposed to a nursery collection. In this example, though only 16 bytes were requested note that the freebytes before GC is zero.

The elements most likely to be of interest are the three copies of the **<tenured>** element describing the occupancy of the heap. These are indicated by the **<tenured tag**.

There are three copies of these elements, to show the state of heap at three important points in time. The first copy shows the state of the heap before collection. The second copy, nested within the **<gc>** element, represents the heap after collection. This shows the amount of live data in the application at the time of the collection. The final copy shows the amount of heap available after the request that triggered the allocation was satisfied. The

difference between this and the amount of free space available immediately after the collection may be more than the actual amount requested. The memory manager allocates memory to threads in chunks to minimize contention on the heap lock.

The nested `<loa>` and `<soa>` elements describe the heap used by the large and small object areas. The large object area is a small area of the heap reserved for large object allocations, while the small object area is the "normal" heap. All objects are initially allocated to the small object area, but if the small area is full, objects larger than 64KB are allocated to the large object area. If the large object area is not required by the application (that is, if the application does not allocate any large objects), the memory management routine quickly shrinks the large object area down to nothing so that the whole heap is available for "normal" allocations.

If present the element `<expansion>` Indicates that during the handling of the allocation (but after the garbage collection), a heap expansion was triggered. The area expanded, the amount by which the area was increased (in bytes), its new size, the time taken to expand, and the reason for the expansion are shown.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Error messages output showing dump events

```
JVMDUMP006I Processing Dump Event "throw", detail
  "java/lang/OutOfMemoryError" - Please Wait.
JVMDUMP007I JVM Requesting Heap Dump using
  'C:\Temp\Dumps\heapdump.20070715.173835.1012.phd'
JVMDUMP010I Heap Dump written to
  C:\Temp\Dumps\heapdump.20070715.173835.1012.phd
JVMDUMP007I JVM Requesting Java Dump using
  'C:\WASv61\profiles\AppSrv01\javacore.20070715.1738
  35.1012.txt'
JVMDUMP010I Java Dump written to
  C:\WASv61\profiles\AppSrv01\javacore.20070715.17383
  5.1012.txt
JVMDUMP013I Processed Dump Event "throw", detail
  "java/lang/OutOfMemoryError"
```

Figure 9-28. Error messages output showing dump events

WA5711.0

Notes:

In this case both a heapdump and a javadump were configured to be created if an OutOfMemory exception was thrown. Search the native_stderr.log output for “Heap Dump” or “Java Dump” and look for the AF entry directly preceding the output.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

VerboseGC output for gencon policy



Figure 9-29. VerboseGC output for gencon policy

WA5711.0

Notes:

As a reminder, gencon divides heap into “nursery” and “tenured” segments providing fast collection for short-lived objects. This provides maximum throughput with minimal pause times.

The example above illustrates a normal allocation failure when using the gencon policy. At the end of GC cycle there space available to allocate the requested bytes of 144. The **id** field indicates that this is the 35th allocation failure and **type** shows this to be a nursery collection as opposed to a tenured collection.

Notice that the heap occupancy before the GC shows there are 0 bytes available in the nursery. The scavenge occurs and **flipped objectcount** shows 1059594 objects were flipped to the survivor space therefor remaining in the new area. The **tenured objectcount** shows 12580 objects were moved into the tenured space. Notice now there are 116754360 bytes now free in the nursery.

Additionally, the **tenureage** shows the number of times an object will “flip” between semi-spaces before being moved to the old/tenured generation. In the example above the age required is 14.

The **tilratio** of 90 means shows that 90% of the nursery is dedicated to the allocate space as opposed to the survivor space.

Instructor notes:

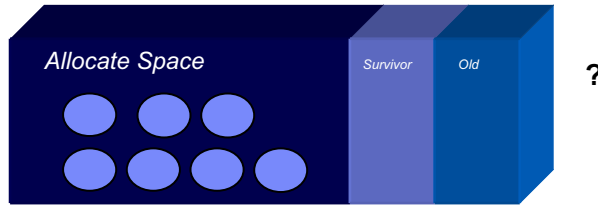
Purpose —

Details —

Additional information —

Transition statement —

VerboseGC output for gencon failures



- Scavenging can fail due to a complete lack of space
 - Abort the scavenge and attempt a global collect

```
<warning details="aborted collection" />
```

The collection is aborted and will result in a global collect

- Nursery collections sometimes determine that they will be insufficient
 - Failure to complete a Nursery collect
 - Insufficient resources

- Collect will be promoted to a Global GC

```
<percolating_collect reason="insufficient remaining tenure space" />
```

Reasoning behind incurring a global collect during a nursery collect

Figure 9-30. VerboseGC output for gencon failures

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Extensible Verbose Tool Kit (EVTK)

- EVTK: Verbose GC visualizer and analyzer
 - Available through the IBM Support Assistant

 - Reduces barrier to understanding verbose output
 - Visualize GC data to track trends and relationships
 - Analyze results and provide general feedback
 - Extend to consume output related to application
 - Build plug-able analyzers for specific application needs

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Extensible Verbose Toolkit overview

- The Extensible Verbose Toolkit (EVTK) is a visualizer for verbose garbage collection output
 - The tool parses and plots verbose GC output and garbage collection traces (-Xtgc output)
- Launched from the IBM Support Assistant
- The EVTK provides
 - Raw view of data
 - Line plots to visualize a variety of GC data characteristics
 - Tabulated reports with heap occupancy recommendations
 - View of multiple datasets on a single set of axes
 - Ability to save data as an image (jpeg) or comma separated file (csv)

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

EVTK usage scenarios

- Investigate performance problems
 - Long periods of pausing or unresponsiveness
- Evaluate your heap size
 - Check heap occupancy and adjust heap size if needed
- Garbage collection policy tuning
 - Examine GC characteristics, compare different policies
- Look for memory growth
 - Heap consumption slowly increasing over time
 - Evaluate the general health of an application

Figure 9-33. EVTK usage scenarios

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Plotting data with the EVTK

Use **File -> Open** to open a new input file

Use **File -> Add** to add multiple input files to a single data set for comparison and aggregated display

Right-click the plot and use the **context menu** to export data

The **VGC Data** menu allows you to choose what data to display

The **Axes** panel supports customized units and pan-and-zoom

The **Line plot** tab contains the data visualization

Figure 9-34. Plotting data with the EVTK

WA5711.0

Notes:

This graph shows the GC cycle pause times for three runs of the same application, all using the gencon GC policy

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Types of graphs

- The EVTK has built-in support for over forty different types of graphs
 - These are configured in the VGC Data menu
 - Options vary depending on the current data set and the parsers and post-processors that are enabled
- Some of the VGC graph types are:
 - Used total heap
 - Pause times (mark-sweep-compact collections)
 - Pause times (totals, including exclusive access)
 - Compact times
 - Weak references cleared
 - Soft references cleared
 - Free tenured heap (after collection)
 - Tenured heap size
 - Tenure age
 - Free LOA (after collection)
 - Free SOA (after collection)
 - Total LOA
 - Total SOA
- **Note:** Different graph types and a different menu are available for TGC output

Figure 9-35. Types of graphs

WA5711.0

Notes:

Instructor notes:

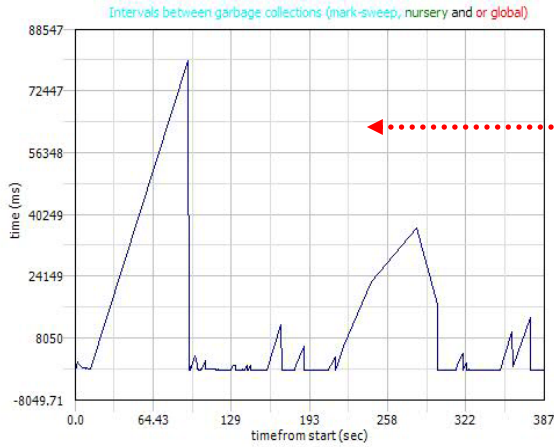
Purpose —

Details —

Additional information —

Transition statement —

Garbage collection trigger graphs



This graph shows intervals between garbage collection cycles; notice that it shrinks to near 0ms for extended periods (horizontal portions of the graph)

Each dot in this graph represents a garbage collection cycle. All of these cycles ran for reason "af" – allocation failure. Notice that the dot concentration lines up with the trigger graph.

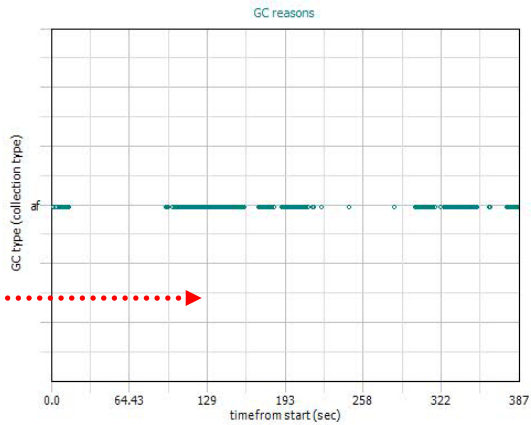


Figure 9-36. Garbage collection trigger graphs

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Heap usage and occupancy recommendation

[Occupancy recommendation](#)

[Summary](#)

Occupancy recommendation

The mean occupancy is 98%. This is a bit high, so you may improve collection times by increasing your heap size.

Summary

Largest memory request (bytes)	1409048
Allocation failure count	563
GC Mode	optthruput
Proportion of time spent in garbage collection pauses (%)	34.99
Concurrent collection count	0
Proportion of time spent unpaused (%)	65.01
Forced collection count	0
Number of collections	298
Mean interval (sec)	0.62
Mean pause (ms)	217

The summary report shows that mean heap occupancy is 98% and that the application is spending over a third of its time doing garbage collection

This graph shows heap usage after garbage collection; it jumps up to around 60M and stays there

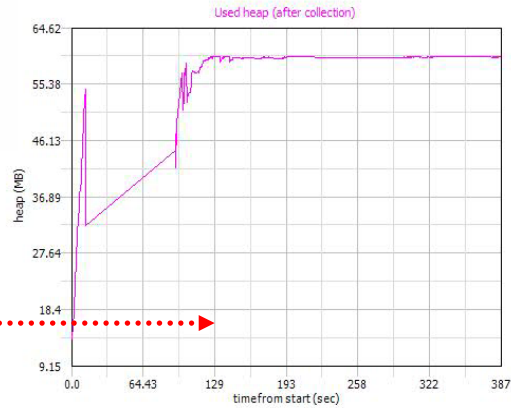


Figure 9-37. Heap usage and occupancy recommendation

WA5711.0

Notes:

Instructor notes:

Purpose —

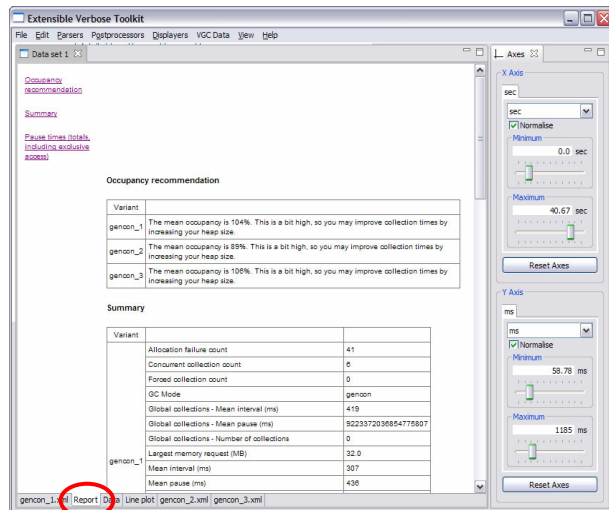
Details —

Additional information —

Transition statement —

Reports and recommendations

- Report contents can be configured using VGC menu options
 - Occupancy recommendations tell you how to adjust heap size for better performance
 - Summary information is generated for each input in the data set
 - Graphs included for all GC display data
- Can export as HTML by right-clicking and using the context menu



The **Report** tab contains the report for the current data set

Figure 9-38. Reports and recommendations

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

IBM Pattern Mapping and Analysis Tool (PMAT)

- Launched from the IBM Support Assistant
- Provides the following:
 - Tabulated and graphical views
 - Summary analysis
 - Analysis includes recommendations
- Works with IBM JVM 1.3.x and up

Figure 9-39. IBM Pattern Mapping and Analysis Tool (PMAT)

WA5711.0

Notes:

PMAT is a tool written by IBM Support that parses VerboseGC output from IBM 1.3.X or later JVMs. PMAT will display all GC events in text, tabulated, and graphical views. One nice feature of PMAT is that it will also interpret the verboseGC information and provide recommendations.

PMAT is available as a tool plug-in within ISA.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

PMAT: Summary page

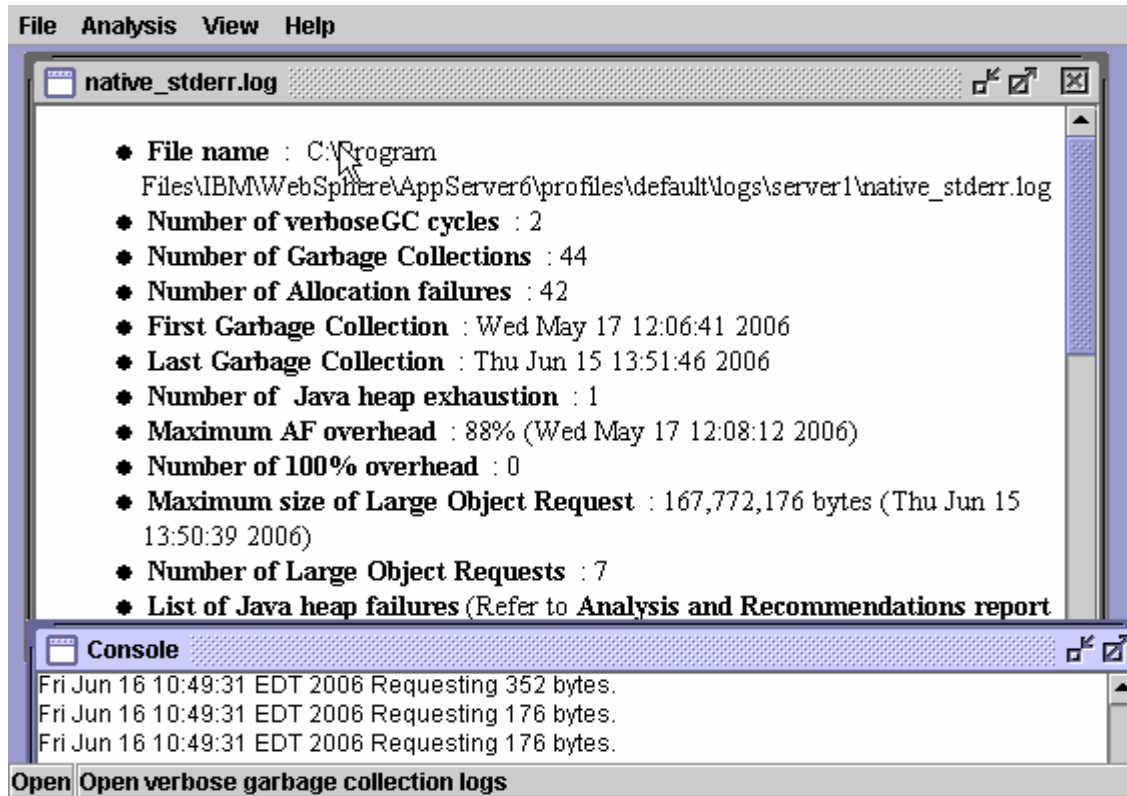


Figure 9-40. PMAT: Summary page

WA5711.0

Notes:

The PMAT tools contains two windows, one contains the summary report and the other displays console events.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

PMAT: Analysis and recommendations

Garbage collection start / finish	Analysis	Recommendations
Wed May 17 12:06:41 2006 Wed May 17 12:08:12 2006	No Java heap exhaustion found	There seems to be a steady increase in Java heap usage. (ratio(%): 162.68823 with percentage error(%): 3.9455755)
Thu Jun 15 13:42:50 2006 Thu Jun 15 13:51:46 2006	Java heap shortage. 167,772,176 bytes requested while 110,256,824 bytes available Thu Jun 15 13:50:39 2006	Increase maximum Java heap size using -Xmx option. If it does not work, review Java heap dump

Fri Jun 16 10:49:31 EDT 2006 Requesting 352 bytes.
 Fri Jun 16 10:49:31 EDT 2006 Requesting 176 bytes.
 Fri Jun 16 10:49:31 EDT 2006 Requesting 176 bytes.

Figure 9-41. PMAT: Analysis and recommendations

WA5711.0

Notes:

If you scroll to the bottom of the summary page, the tool provides recommendations. In this example the tool first suggests increasing the Java heap, and if that fails it instructs the user to review a Java heap dump.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

PMAT: Chart view

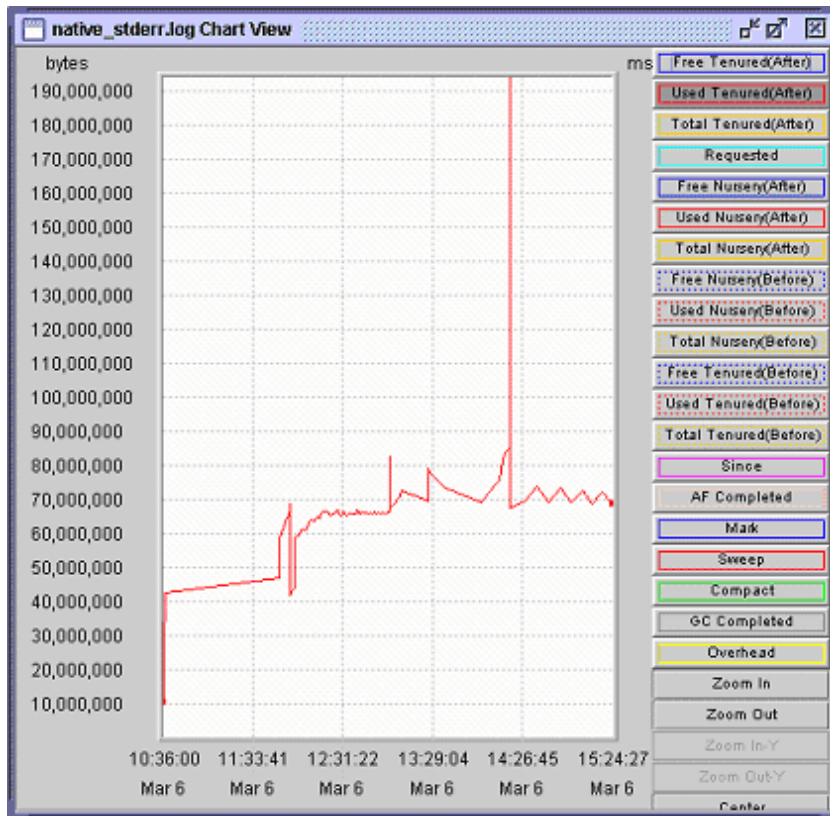


Figure 9-42. PMAT: Chart view

WA5711.0

Notes:

As pictured, the used heap space is graphed. Notice that other data types can be added to the chart as desired. Select the desired data from the panel on the left.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Analyze Java heap dumps

After completing this topic, you should be able to:

- Understand what to expect from data within the heap dump
- Use tools to help process Java heap dump

Figure 9-43. Analyze Java heap dumps

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

OutOfMemory: Interpret heap dumps

- Heapdump
 - Use tools to parse the object trees
 - Almost impossible to do manually
 - Look for root objects holding significant memory resources
 - Includes memory allocated for the root object themselves as well as all the memory required for all the objects in the reference tree
 - Check reference tree for sudden drop in memory
 - Expand the reference tree, and examine each object as you traverse downwards
 - Compare the memory with the object right above it in the tree
 - A significant drop in memory indicates a potential memory leak
- JMap
 - Contact support to parse output
 - Manually inspect the output list
 - Difficult and time consuming

Figure 9-44. OutOfMemory: Interpret heap dumps

WA5711.0

Notes:

The most important diagnostic information for debugging memory leaks is a dump of the JVM heap. With proper tools, you can inspect the objects allocated in the JVM heap, traverse the object tree, and look for potential memory leak candidates. You do this by searching for root objects holding a significant portion of the memory in the heap, and traverse the object tree until you find a location where the memory drops significantly from parent to child. When you find such a drop in memory, the parent becomes a memory leak suspect.

There are two types of output that are useful to IBM Support. The first is the IBM formatted heap dump file (php). You can use the MDD4J tool to parse and display the information within the heap dumps. The other type of output is the Sun JMAP output. Parsing JMAP output is difficult and time consuming, and requires IBM development assistance.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

How to analyze a Java heap dump

- Single dump analysis
 - Inspects a single heap dump file for container objects that have very large reach size compared to their largest child object
 - Most commonly used to analyze automatically generated heap dumps after an `OutOfMemoryException`
- Comparative analysis
 - Analyzes heap growth between two dumps taken over time
 - Identifies the objects with the largest size difference and their ownership relationships
- Analysis results
 - Summary of analysis results showing heap contents, size and growth
 - Lists suspected data structures, data types and packages contributing to the growth in heap usage
 - Tabular views of all the objects and data-types in the memory dump with filters and sorted columns

Figure 9-45. How to analyze a Java heap dump

WA5711.0

Notes:

You will most often use the first option to analyze a heap dump that was automatically generated at the time of an `OutOfMemoryException`. This method looks for objects in the heap that have very large reach sizes compared to their largest child objects.

The comparative analysis feature examines the changes in the heap between two heap dumps taken over time to analyze which objects have grown the most. If you believe an application to be leaking memory, a comparative analysis of two dumps separated by several minutes will often highlight the objects that are the most likely source of leaking memory, helping you to quickly locate parts of the application code that should be examined.

The analysis results panel displays the results of memory analysis in several ways. It displays first a list of suspected data structures, which are the objects in your heap that are most likely the cause of memory growth. The data in the analysis results screen can be easily sorted and filtered to make it easy to locate objects that interest you.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Memory Dump Diagnostic Tool for Java (MDD4J)

- Memory Dump Diagnostic Tool for Java
 - Available from the **Tools** tab in ISA
- Analyzes Java memory (heap) dumps
 - Identifies data structures that are likely causes of memory growth (leaks) and their relationships
 - Comparison of primary and secondary heap dumps
 - Provides a graphical display of the Java object tree
- Handles several dump formats
 - IBM portable heap dump (.phd)
 - IBM text heap dump (.txt)
 - HPROF dump (hprof.txt)
 - z/OS SVC dump (dump.bin)

Figure 9-46. Memory Dump Diagnostic Tool for Java (MDD4J)

WA5711.0

Notes:

The MDD4J tool provides good analysis and interpretation of phd files. There are two forms of analysis that can be performed. The first is the analysis of a single heap dump from a single OutOfMemoryError failure. The second is a comparison of a primary (or failure) and secondary (or baseline) heap dumps. This is useful when you are unable to determine the leaking object from the analysis of a single heapdump from the OutOfMemory failure. The baseline heap dump is provided from a JVM running in a healthy state, before the leak has consumed significant memory resources. You then compare a failure heap dump from the baseline to determine what the change in allocated objects is.

You are focused on the analysis of primary heap dump files. The MDD4J tool will parse the heapdump and provide the results to the user. The first information of interest is the list of memory leak suspects. The tech preview version of MDD4J lists a single suspect, and the subsequent release will list five potential suspects.

The tool also displays the data graphically and within a traversable object tree.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

MDD4J Analysis Summary tab

Summary and Next Steps

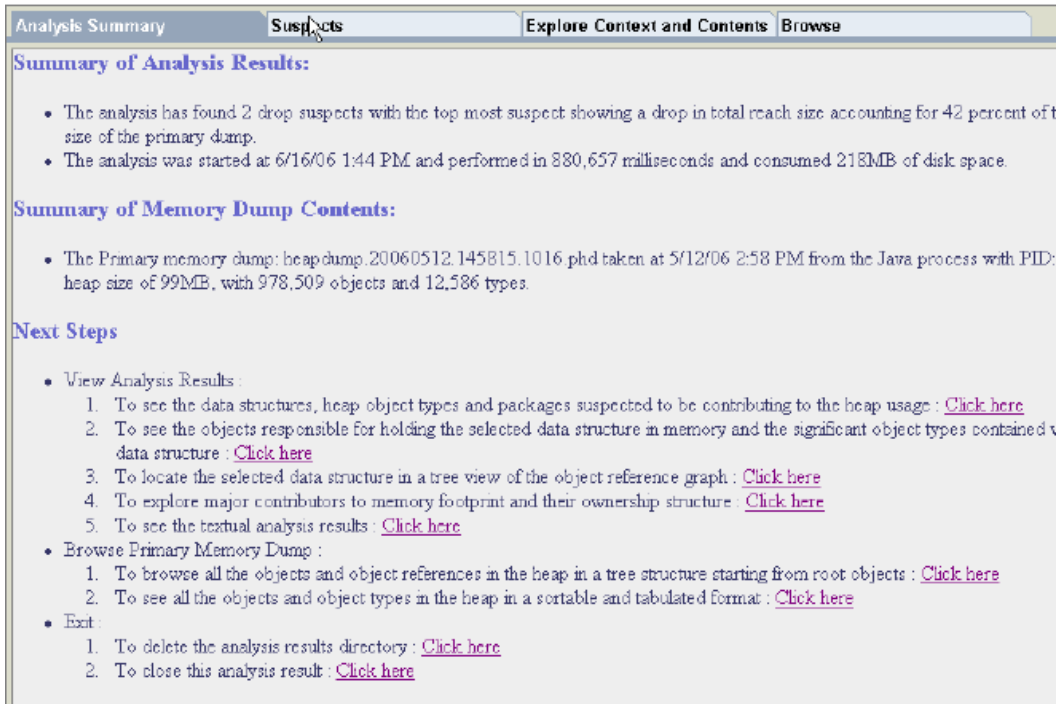


Figure 9-47. MDD4J Analysis Summary tab

WA5711.0

Notes:

The **Analysis Summary** page gives of a quick summary information extracted from the heap dump. This information includes the size of the memory dumped to file, and the number of objects contained in memory. Its important to verify that the heap size dumped to file is correct, otherwise you are looking at truncated heap dump which will not contain useful information.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

MDD4J Suspects tab

- The usual suspects: data structures, object types, packages

The screenshot shows the MDD4J Suspects tab with the following content:

Data structures with large drops in reach size

The following table shows objects in the primary heap dump with a large drop in reach sizes from the parent object to its largest reach object. The reach size of an object is calculated by adding the sizes of all objects which are reachable from that object during a single the object references in the heap. These parent objects with large drops can be indicative of array based container objects holding on number of growing child objects. Click on each row to explore the context and contents of the encapsulating data structure.

#	Object type of suspected container	Reach size of the container object	Drop in reach size
0	java/lang/Object[]	42MB	42MB
1	com/ibm/ws/management/event/NotificationService	84MB	24MB

Object Types that contribute most to heap size

#	Suspected Object Type	Number of instances	Bytes
0	java/lang/String	253,672	7,102,816
1	char[]	247,718	25,614,366
2	java/util/HashMap\$Entry	137,789	3,858,092
3	java/util/HashMap\$Entry[]	26,073	2,641,348
4	java/util/HashMap	19,818	951,264

Packages that contribute most to heap size

#	Suspected Package	Number of instances
0	java/lang	293,857
1	java/util	229,550

Figure 9-48. MDD4J Suspects tab

WA5711.0

Notes:

The **Suspects** tab is where MDD4J shows you who is potentially leaking the memory. The result provided by the current technical preview version of MDD4J only shows one potential suspect. In most cases the first suspect identified will not be the leaking object. In the next version, the top five suspects will be displayed, giving you much more reliable information.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

MDD4J Explore Context and Contents tab

- Shows ownership context of select suspects

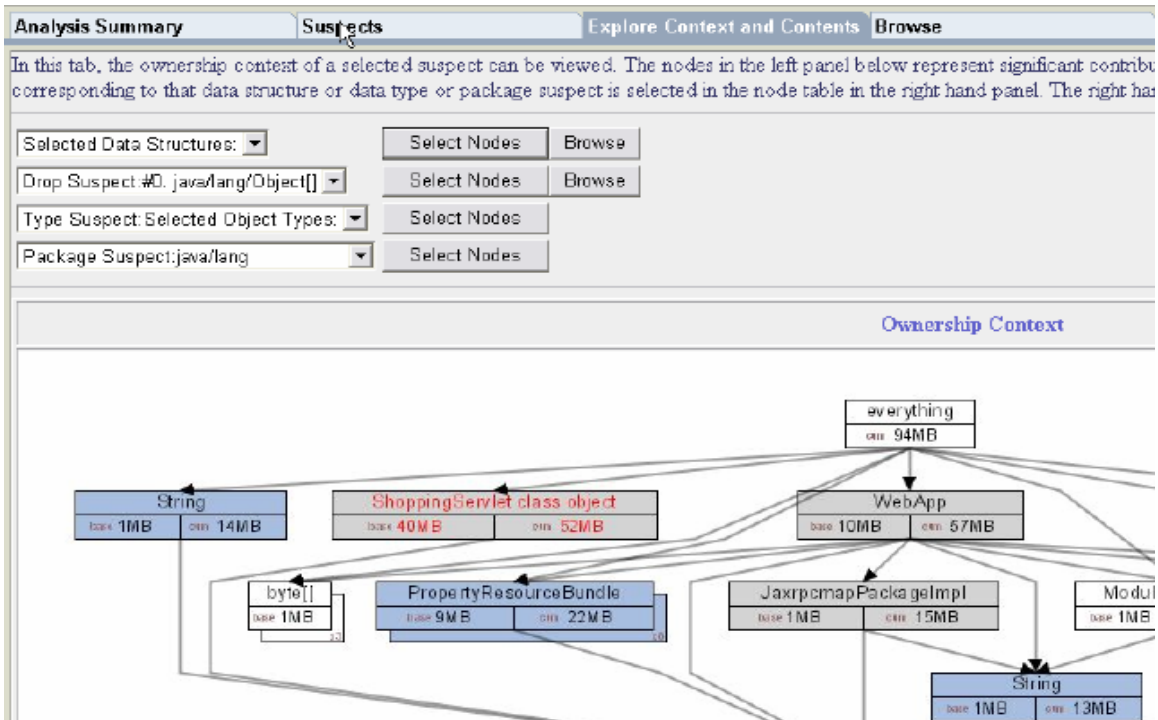


Figure 9-49. MDD4J Explore Context and Contents tab

WA5711.0

Notes:

The **Explore Context and Contents** tab shows the ownership context of selected suspects. Users can select nodes listed on the drop-down list, and graphically explore the ownership context to identify objects consuming significant memory.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

MDD4J Browse tab

- Traverse the object tree

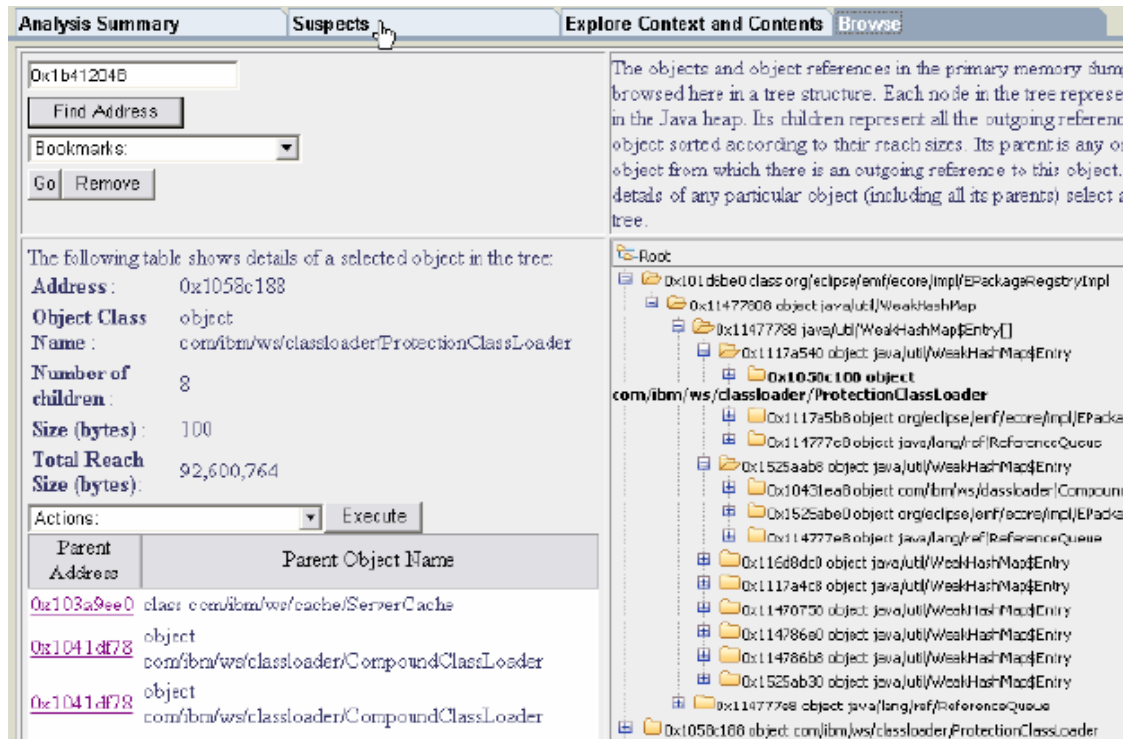


Figure 9-50. MDD4J Browse tab

WA5711.0

Notes:

The **Browse** tab allows you to traverse the object tree looking for significant drops in memory usage. On the left, you can see the details of the highlighted object in the tree on the right. The key information is the Total Reach Size, which tells you how much memory is being used by the highlighted object, as well as all of the referenced objects below it in the tree. You can identify the leaking object by traversing down the tree until you go from a parent to one of its children, and the Total Reach Size drops significantly.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. List three causes of Java heap OutOfMemory error.
2. What JVM-related statistics can be monitored with the PMI Basic setting?
3. What diagnostic data does verboseGC output provide for identifying an OutOfMemory condition?

Figure 9-51. Checkpoint

WA5711.0

Notes:

Write down your answers here:

- 1.
- 2.
- 3.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

1. List three causes of Java heap OutOfMemory error.
 1. **Insufficient maximum heap size for user load**
 2. **Memory leak in the Java code**
 3. **Memory leak in native code**

2. What JVM-related statistics can be monitored with the PMI Basic setting?
 1. **JVM UpTime**
 2. **JVM HeapSize**
 3. **JVM UsedMemory**

3. What diagnostic data does verboseGC output provide for identifying an OutOfMemory condition?
 1. **Size of object being allocated**
 2. **Available memory on the Java heap**
 3. **JVM response to an allocation request**

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Some useful Web addresses

- IBM JVM Diagnosis documentation:
 - <http://www.ibm.com/developerworks/java/jdk/diagnosis/>
- IBM JRE and JDK forum:
 - http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=367&start=0&thRange=15&cat=10
- Memory leak detection and analysis in WebSphere Application Server
 - http://www.ibm.com/developerworks/websphere/library/techarticles/0606_poddar/0606_poddar.html
- Garbage collection policy and tuning article on developerWorks
 - <http://www.ibm.com/developerworks/java/library/j-ibmjava3/>

Figure 9-53. Some useful Web addresses

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe what causes OutOfMemory conditions
- Use the Tivoli Performance Viewer and javacore files to detect OutOfMemoryConditions
- Obtain a VerboseGC log
- Obtain a Java heap dump
- Interpret a verboseGC log
- Analyze a Java heap dump

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 4: Troubleshooting an out-of-memory condition

Figure 9-55. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

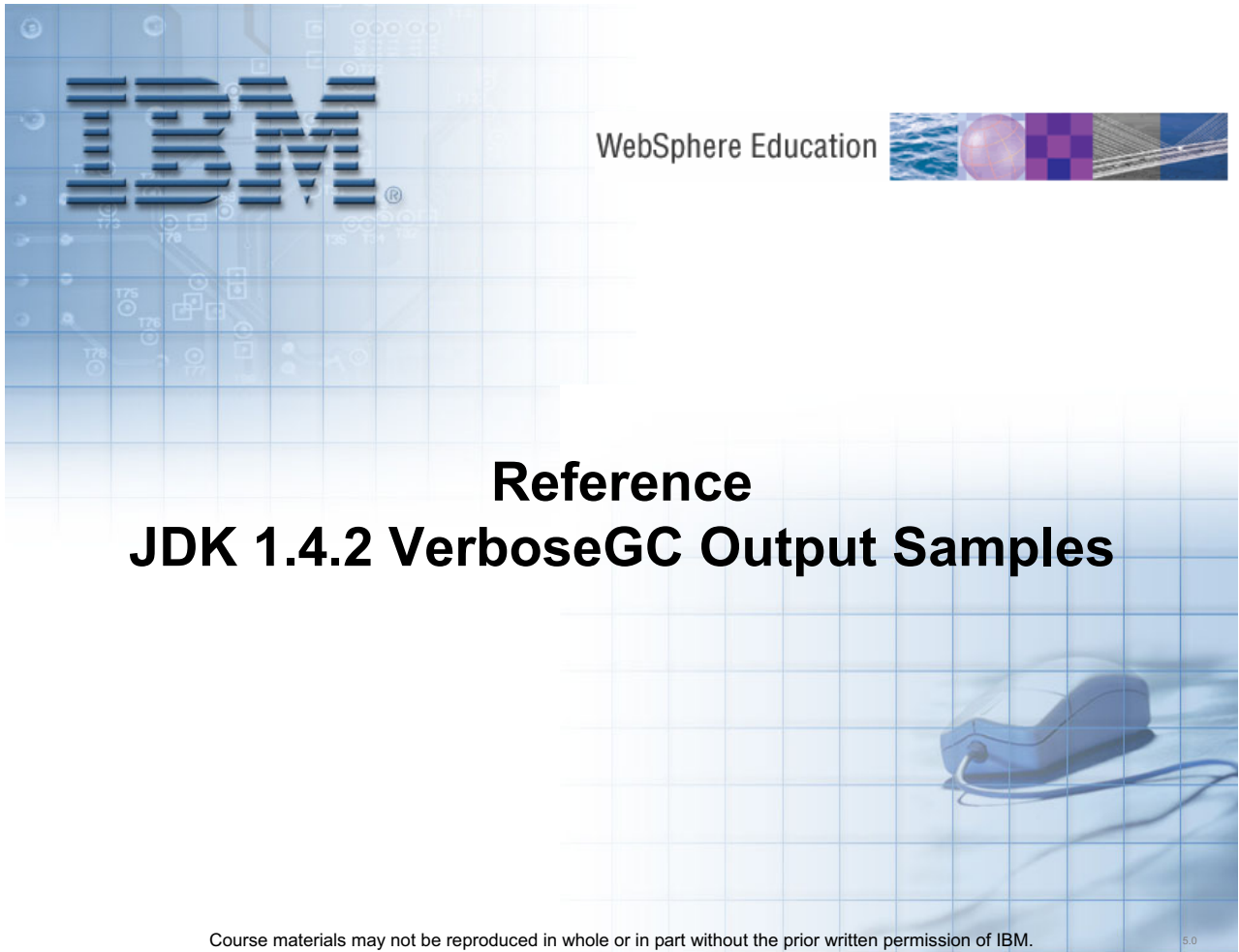


Figure 9-56. Reference JDK 1.4.2 VerboseGC Output Samples

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Sample VerboseGC output: Allocation failure

```
<AF[15]: Allocation Failure. need 10485776 bytes, 39 ms
since last AF>
<AF[15]: managing allocation failure, action=2
(43700440/85129728)>
  <GC(15): freeing class
sun.reflect.GeneratedMethodAccessor8(105962D8)>
  <GC(15): unloaded and freed 1 class>
  <GC(15): GC cycle started Wed May 17 12:08:11 2006
  <GC(15): freed 3449536 bytes, 55% free
(47149976/85129728), in 184 ms>
  <GC(15): mark: 171 ms, sweep: 13 ms, compact: 0 ms>
  <GC(15): refs: soft 0 (age >= 6), weak 0, final 0,
phantom 0>
<AF[15]: completed in 185 ms>
```

Figure 9-57. Sample VerboseGC output: Allocation failure

WA5711.0

Notes:

Here is some sample verboseGC output generated by the PlantsByWebSphere application. This sample shows a single allocation failure, and the subsequent garbage collection.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — The output shows you important details about the available memory in the heap and the time it takes to free additional heap space.

How to interpret VerboseGC output

- Line 1:** <AF[15]: Allocation Failure. need 10485776 bytes, 39 ms since last AF>
- How many allocation failures have occurred
 - How many bytes are needed
 - Time since last allocation failure
- Line 2:** <AF[15]: managing allocation failure, action=2 (43700440/85129728)>
- How much free space is available in the heap
- Line 5:** <GC(15): GC cycle started Wed May 17 12:08:11 2006
- Shows the start and timestamp of garbage collection event
- Line 6:** <GC(15): freed 3449536 bytes, 55% free (47149976/85129728), in 184 ms>
- How many bytes were freed, and what percentage of the heap is free
 - How long it took for garbage collection to run
- Line 9:** <AF[15]: completed in 185 ms>
- Total amount of time it took to process the allocation failure

Figure 9-58. How to interpret VerboseGC output

WA5711.0

Notes:

Line 1 is the first line of the allocation failure. The number in brackets is the current count of allocation failures. This line also shows you how many bytes are needed in order to satisfy the memory requirement for the object allocation request, and the time in milliseconds since the last allocation failure. On line 2, it gets more detailed information about the current state of the Java heap. The information in parenthesis indicates that total heap size is 85,129,728 bytes free, and of that 43,700,440 bytes are available. In this case, there is not 10,485,776 bytes of contiguous space available, so garbage collection is started.

Lines 5 and 6 show details of the garbage collection. On line 5, garbage collection starts at 12:08:11 on May 17, 2006. On line 6, GC has completed after freeing 3,449,536 bytes in 184 ms. At this time, 47,149,976 bytes of the 85,129,728 byte heap, or 55% of the heap, is now free.

Finally, on line 9, it shows that it took a total of 185 ms to process allocation failure 15.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Sample VerboseGC output: OutOfMemory (1 of 3)

```

<AF[26]: Allocation Failure. need 167772176 bytes, 55031
ms since last AF>
<AF[26]: managing allocation failure, action=2
(49959168/256637440)>
  <GC(26): GC cycle started Thu Jun 15 13:50:39 2006
  <GC(26): freed 60297656 bytes, 42% free
(110256824/256637440), in 1442 ms>
  <GC(26): mark: 704 ms, sweep: 16 ms, compact: 722 ms>
  <GC(26): refs: soft 200 (age >= 4), weak 0, final 146,
phantom 10>
  <GC(26): moved 314304 objects, 23247400 bytes,
reason=1, used 48 more bytes>
<AF[26]: managing allocation failure, action=3
(110256824/256637440)>
  <GC(26): need to expand mark bits for 268433920-byte
heap>
  <GC(26): expanded mark bits by 184320 to 4194304 bytes>

```

Figure 9-59. Sample VerboseGC output: OutOfMemory (1 of 3)

WA5711.0

Notes:

Here is another example that shows the processing of an OutOfMemory condition. The verbose garbage collection output comes from an IBM 1.4.2 JVM. The output will vary for other JVMs, but they will generally produce the same type of information.

The first remark in bold in the verbose garbage collection output indicates that there was an allocation failure trying to allocate an object that is requesting 167.7 MB of memory. The second line in bold shows the state of the Java heap at that time. The heap size is 256 MB, of which only about 50 MB of memory is free. The third line in bold shows that the GC cycle freed about 60 MB giving a total of 110 MB, or 42% of the 256 MB Java heap, available for allocation. This is still insufficient for an allocation request of 167.7, so in the 4th line in bold the output shows that the garbage collector is starting to manage the allocation failure. The first thing that the garbage collector does is request that the JVM memory management unit expand the heap. Here you can see in the fifth line in bold that the JVM is attempting to expand the heap to 268 MB, which in this case happens to be the maximum value.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Sample VerboseGC output: OutOfMemory (2 of 3)

```
<GC(26): need to expand alloc bits for 268433920-byte
heap>
  <GC(26): expanded alloc bits by 184320 to 4194304
bytes>
  <GC(26): need to expand FR bits for 268433920-byte
heap>
  <GC(26): expanded FR bits by 368640 to 8388608 bytes>
  <GC(26): expanded heap fully by 11796480 to 268433920
bytes, 45% free>
<AF[26]: managing allocation failure, action=4
(122053304/268433920)>
<AF[26]: managing allocation failure, action=6
(122053304/268433920)>
JVMDG217: Dump Handler is Processing OutOfMemory - Please
Wait.
JVMDG315: JVM Requesting Heap dump file
```

Figure 9-60. Sample VerboseGC output: OutOfMemory (2 of 3)

WA5711.0

Notes:

Here you can see in the first line in bold that the JVM managed to expand the heap by about 12 megabytes to 268 MB, and 45% of it is free. On the next line, though, you see that the total is still only about 122 MB, which still is not enough to satisfy the 167 MB request. The allocation failure routine then begins OutOfMemory processing. On the last line you see the JVM getting ready to produce a heap dump file.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Sample VerboseGC output: OutOfMemory (3 of 3)

```
JVMDG318: Heap dump file written to C:\Program
Files\IBM\WebSphere\AppServer6\profiles\default\heapdump.
20060615.135039.1448.phd
JVMDG303: JVM Requesting Java core file
JVMDG304: Java core file written to C:\Program
Files\IBM\WebSphere\AppServer6\profiles\default\javacore.
20060615.135054.1448.txt
JVMDG274: Dump Handler has Processed OutOfMemory.
<AF[26]: Insufficient space in Javaheap to satisfy
allocation request>
<AF[26]: completed in 18805 ms>
```

Figure 9-61. Sample VerboseGC output: OutOfMemory (3 of 3)

WA5711.0

Notes:

The verboseGC output shows you where the heap dump file is written to disk. The output then shows the JVM dumping a javacore file, and also identifies the location on disk.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — There are tools available that will parse the verboseGC information for you, and one of those tools is PMAT.

Unit 10.Database connection and configuration problems

Estimated time

00:30

What this unit is about

This unit investigates the common problems that can arise when creating a connection to a database and the techniques used to solve them.

What you should be able to do

After completing this unit, you should be able to:

- Describe the major functions of the JDBC provider and the data source
- Install and configure the JDBC provider and the data source
- Explain the concept of the WebSphere Application Server Test Connection service and use it to verify that the database connection is correctly configured
- Detect database connection problems and correct them

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

WebSphere Application Server Network Deployment, Version 6.1.x information center - Configuring a JDBC provider and data source:

http://publib.boulder.ibm.com/info center/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tdat_tcrtprovds.html

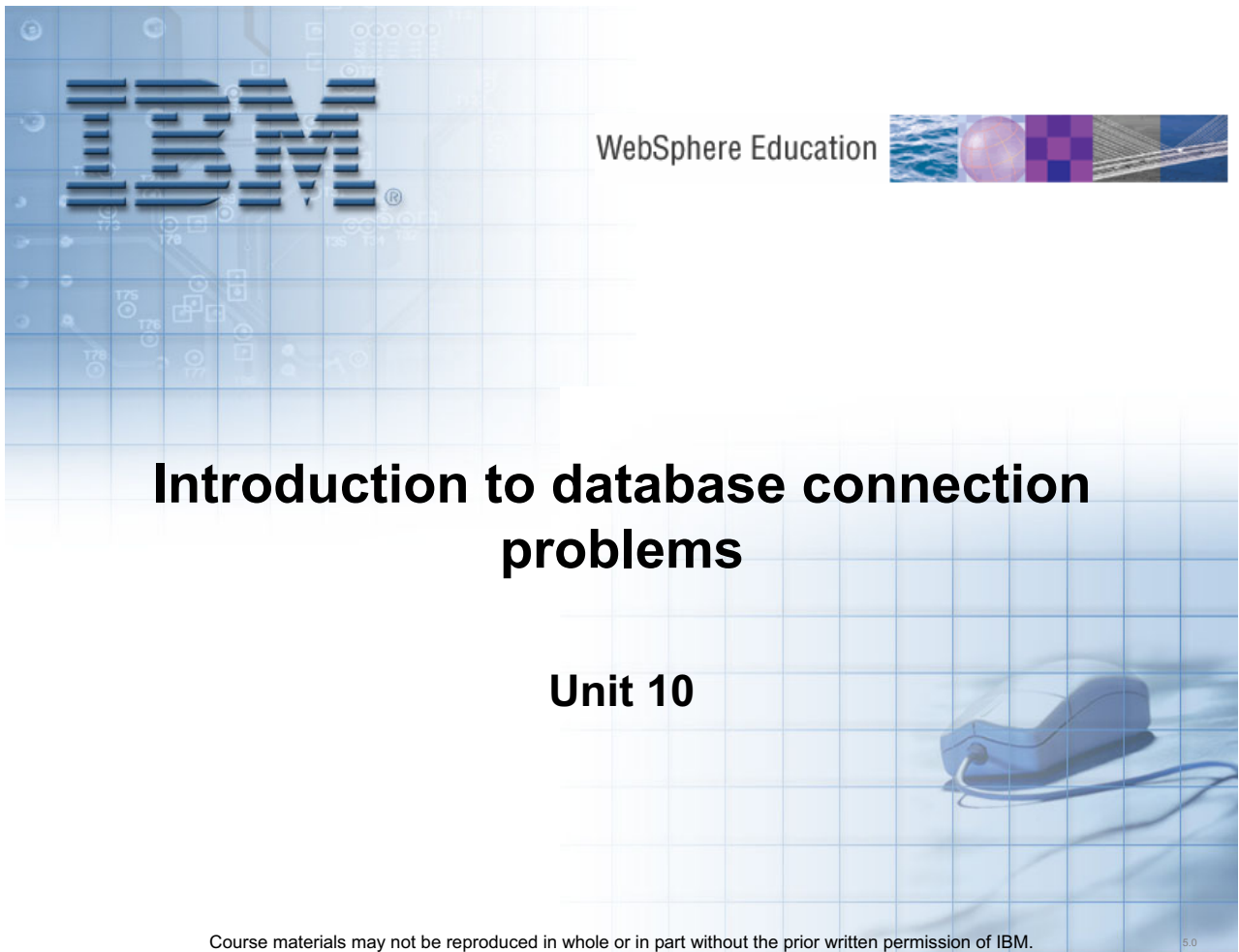


Figure 10-1. Introduction to database connection problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — This module is not intended as a tutorial on how to install and configure a database connection. However, in order to troubleshoot a connection problem it is necessary to go through the installation and configuration process.

Estimated time

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe the role of the JDBC provider
- Describe the role of the data source
- Understand some of the common problems encountered when configuring database connections
- Test a database connection using the WebSphere Application Server Version 6.1 *Test Connection* service
- Troubleshoot some of the common problems encountered when testing database connections

Figure 10-2. Unit objectives

WA5711.0

Notes:

WebSphere Application Server V6.1 implements the Java Database Connectivity (JDBC) specification, providing a standards-based interface to the compliant JDBC driver implementation of any vendor. The driver is represented in WebSphere Application Server V6.1 as a JDBC provider. At least one data source is associated with each JDBC provider and can be configured through the WebSphere Application Server V6.1 administrative console. While creating a JDBC connection is typically a simple task, problems can arise around the definition of the JDBC provider or the parameters required to create the data source. Prior to deploying applications that leverage the database connection, the data source can be verified using the *Test Connection* service of WebSphere Application Server V6.1.

JDBC providers and JDBC data sources in WebSphere Application Server V6.1 have changed very little since WebSphere Application Server V6.0. The noticeable changes include:

- Enhanced Performance Monitoring Infrastructure (PMI) support

- The Cloudscape test database is now the product of the open source Apache Derby project. As a result, the Cloudscape 10.1.x JDBC providers are based on the Derby JDBC classes

Instructor notes:

Purpose — Understand the roles of the JDBC provider and data source in a WebSphere Application Server V6.1 database connection and verify the correctness of the database connection through the Test Connection service

Details — In order to successfully create a database connection, the user needs to configure a JDBC provider which describes the JDBC driver implementation, and create a data source that represents the actual database that users will access through the connection.

Additional information — The WebSphere Application Server V6.1 information center is a good source of information

Transition statement — Look at the role the JDBC provider and the data source in creating a database connection. What are some of the common problems that might arise during database connection creation?

The role of the JDBC provider

- The JDBC provider object supplies the specific JDBC driver implementation class for access to a specific vendor database
- The driver implementation classes are supplied by defining the WebSphere variables associated with the provider
- Java classes and native libraries can be specified by the following:
 - DB2UNIVERSAL_JDBC_DRIVER_PATH
 - INFORMIX_JDBC_DRIVER_PATH
 - DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH
 - ORACLE_JDBC_DRIVER_PATH
 - And so on
- Custom variables can be defined in WebSphere Application Server in order to accommodate any vendor-specific driver implementation

Figure 10-3. The role of the JDBC provider

WA5711.0

Notes:

The JDBC provider represents the vendor-specific database driver implementation. The WebSphere variables reference both Java and native libraries. JDBC type 1 and 2 drivers require native libraries in order to function. Therefore, it will be necessary to reference one or more native libraries. JDBC type 3 and 4 drivers are pure Java drivers and only require JAR files to be referenced by the WebSphere variables. For example, creating a type 4 DB2 Universal JDBC driver relies on the DB2 JAR files for the implementation classes. In order to configure this driver, the `DB2_UNIVERSAL_DRIVER_PATH` and `UNIVERSAL_DRIVER_PATH` WebSphere variables will need to be specified in terms of the path to the DB2 JAR files that contain these classes.



Important

If you are in a network deployment distributed environment, always create variables in terms of other variables. For example, if your `derby.jar` file resides in the `derby/lib` directory

of your application server installation, use the DERBY_JDBC_DRIVER_PATH environment variable for the JDBC provider, rather than \${WAS_INSTALL_ROOT}/derby/lib

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — Once the JDBC provider has been defined, at least one data source will need to be created that uses the driver implementation to access the vendor database.

JDBC provider screen

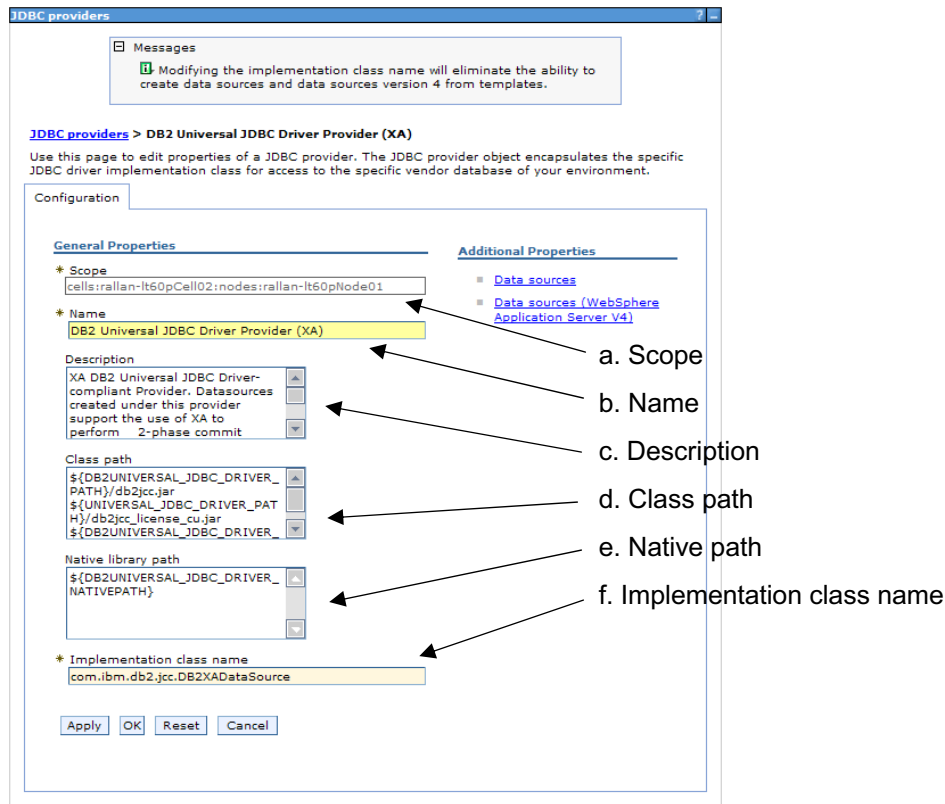


Figure 10-4. JDBC provider screen

WA5711.0

Notes:

The JDBC provider screen can be accessed through the WebSphere Application Server V6.1 administrative console by navigating to **Resources -> JDBC -> JDBC Providers -> JDBC Provider name**. This is the panel you use to specify the name, description, any JAR file references, any native library references and the implementation class name for the provider.

The properties of the JDBC provider screen include:

1. **Scope** - The namespace of the JDBC provider (cell, node, or server scope)
2. **Name** - The display name of this JDBC provider
3. **Description** - A user friendly description
4. **Class path** - The Java *CLASSPATH* addition
5. **Native path** - The system *PATH* addition
6. **Implementation class file name** - The vendor specific Java class that implements the JDBC provider

Instructor notes:

Purpose —

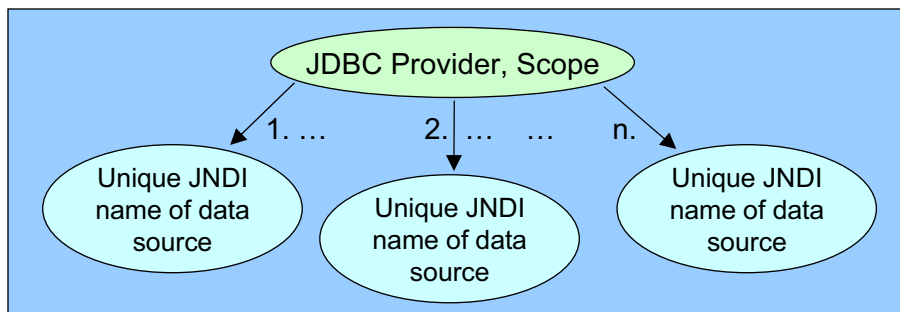
Details —

Additional information —

Transition statement —

The role of the data source

- The *JNDI* name represents the runtime binding of the data source to the JDBC provider in a given scope
 - A data source is analogous to a J2EE Connector Architecture (JCA) connection factory



- Runtime binding creates a highly flexible environment but if problems exist, the symptoms will not be seen until run time
 - The *Test connection* facility of the WebSphere Application Server V6.1 is helpful in uncovering such errors during the deployment phase

Figure 10-5. The role of the data source

WA5711.0

Notes:

A data source requires information that includes:

- The server name
- The listening port of the database server
- The user name credentials
- The type of driver used for the connection

The data source is bound to the runtime environment through its JNDI name. A JDBC provider and the scope has a one-to-many relationship with the JNDI name of the data source. In WebSphere Application Server V6.1 the data source represents the actual connection to the database and uses the JDBC provider implementation to connect to the database system. The data source relies on connection parameters such as the database name, the database server name, the type of the driver being used, the database port number, and the authentication credentials used to determine authorities and privileges for the connection. The authentication credentials are specified using a Java 2 Connection

Architecture (J2C) data entry. This data entry is then specified during the creation of the data source and is used for all subsequent authentication.

User credentials are defined in the **JAAS – J2C authentication data** panel and related to the data source using a JNDI name. This allows credentials to be bound at run time, in the same manner as the data source, resulting in a highly flexible environment. As with the data source, potential problems will not be detected until the references are exercised and JNDI resolution is attempted.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Data source screen (top)

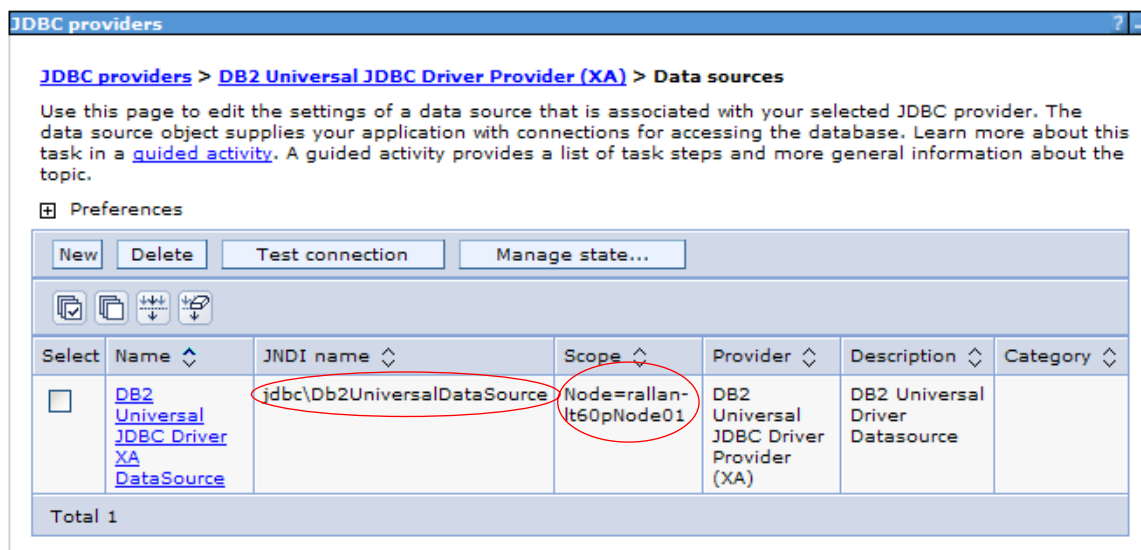


Figure 10-6. Data source screen (top)

WA5711.0

Notes:

The **Data source** screen in the administrative console contains parameter fields for configuring the database connection. At the top of the screen is the **Test connection** button for testing the connection after configuration is complete. The parameter fields include the scope of the data source, the name of the data source, and the JNDI name of the data source which is used by the application to reference the data source. There is also a check box for specifying whether to use this data source in container managed persistence, and a description, category, and widgets for describing a data store helper class.

The **Data sources** panel contains the *Test connection* facility. The parameters that affect the connections success include the *JNDI name*, the *Scope*, and the JCA authentication data (not highlighted in the image).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Creating a database connection: Common problems

- Typical problems that arise when configuring a database connection are:
 - Spelling errors or typos when filling in the parameter values
 - Client JAR files necessary to the JDBC Provider are not present
 - Lack of user permissions on the database server
 - Connectivity problems because of network topology or a database that has not been started
- The problems encountered when configuring a database connection are usually easy to fix if you know what to look for and where to look:
 - WebSphere Application Server V6.1 provides immediate feedback when testing a connection
 - The SystemErr.log contains explicit information about the actual exception

Figure 10-7. Creating a database connection: Common problems

WA5711.0

Notes:

Spelling any of the database connection properties incorrectly, or violating any of the case sensitivity rules of a parameter will result in failure when verifying the connection with the *Test Connection* service. Incorrectly creating a WebSphere variable or neglecting to make the necessary JAR or native libraries available to the connection will also lead to failure.

The data source parameters are focused on the actual database access information such as the database name, server name, port number and access credentials. The database values can usually be verified through a Telnet session to the actual database server as follows:

Simple Telnet test

1. Telnet into the machine using the server name value you are using for the data source.
2. Start a database session using the JCA credentials.
3. Perform a sample of the operations that the application will perform, such as a simple query.
4. Disconnect from the machine.

If you are successful doing this simple test, then the data source parameters are probably correct. However, keep in mind that some differences can arise such as using a server short name as opposed to the fully-qualified domain name. If the application server is on a different machine than where you are performing the telnet test, host resolution issues could occur. Also, the telnet test does not exercise the port number parameter. If this is incorrect, then the data source will fail. You can verify the port number that the database is listening on by exporting the instance parameters. This assumes that you have the authority to do so. If not, you will need to contact your database administrator and ask him to confirm the parameters for you.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using the Test Connection service

- Once the JDBC Provider and the data source have been configured, navigate to **JDBC Providers -> data source name -> Data sources** and click the **Test connection** button
- The **Test connection** button is also available on the actual data source configuration panel of the administrative console

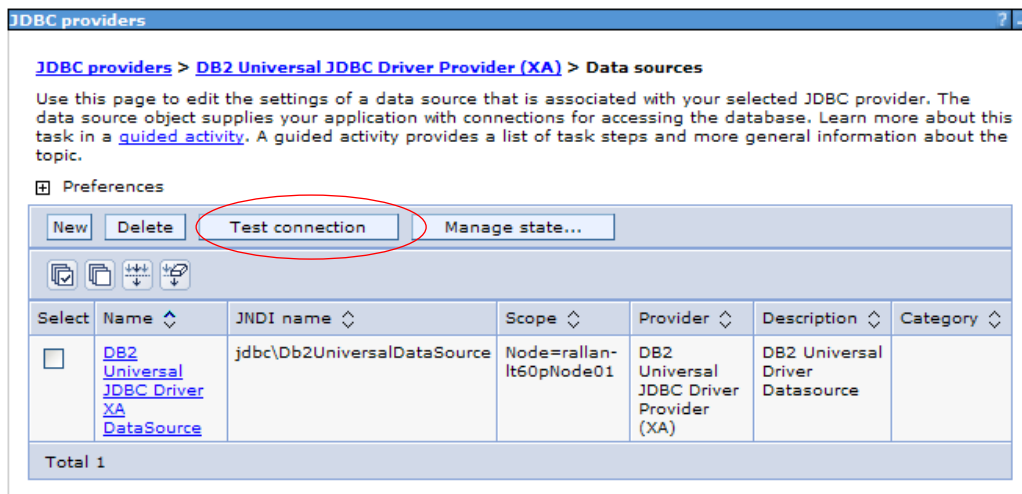


Figure 10-8. Using the Test Connection service

WA5711.0

Notes:

The **Test connection** button is located on two different panels:

1. The **Resources -> JDBC -> JDBC Providers -> Provider name -> Data sources** panel
2. The **Resources -> JDBC -> JDBC Providers -> Provider name -> Data sources -> data source name** panel

The *Test Connection* service uses a default general query of the form `SELECT 1 FROM TABLE1`. This default query string is located in the administrative console panel under **JDBC Providers -> Provider Name -> Data sources -> Data source name -> WebSphere Application Server data source properties** in the **Pretest SQL string** text field. Obviously, this default query can be changed to any correct SQL query.

Keep in mind that the data source cannot be tested until any updated values have been applied and saved to the application server configuration files.

Instructor notes:

Purpose —

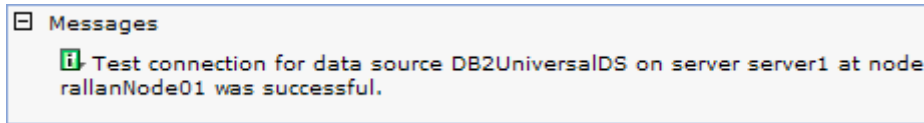
Details —

Additional information —

Transition statement —

Using the Test Connection service (continued)

- If the data source is configured properly, you will receive a message similar to the following:



- If a failure message occurs, then it is time to track down the cause of the failure and resolve it
- A failure could be caused for a number of reasons, and is the subject of the next discussion

Figure 10-9. Using the Test Connection service (continued)

WA5711.0

Notes:

WebSphere Application Server Version V6.1 provides feedback to the administrative console as soon as the *Test Connection* service completes. The failure messages are usually specific enough to lead to a quick and simple resolution. However, some of the messages report on a side effect of the issue rather than the root cause. In those cases, deeper investigation is necessary in order to track down the root cause of the issue.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JDBC provider configuration problems: Other considerations

- If this is a deployment manager scenario, verify that the variables are defined in terms of other variables, such as WAS_INSTALL_ROOT for the application server install path.
- **Important:** Make sure that the scope of the variables you are defining are visible to the variables used in the JDBC Provider. Whether they are Cell, Node, or Server scope could be significant.

Figure 10-10. JDBC provider configuration problems: Other considerations

WA5711.0

Notes:

Defining the WebSphere driver variables in terms of other application server variables allows each node to resolve the variable in the context of its particular environment. Therefore, JAR and native libraries required by the JDBC provider can reside locally, on each node, in a directory relative to the installation path of each node in the cell.

The scope in which the variables, JDBC provider and data source are defined is extremely important to the visibility of each component. Defining the driver implementation at the server level will lead to connectivity failure if you attempt to use the data source at the node level. These driver level variables are not visible at the node scope and will result in class loader exceptions because the system cannot locate the necessary JAR or native libraries.

Instructor notes:

Purpose —

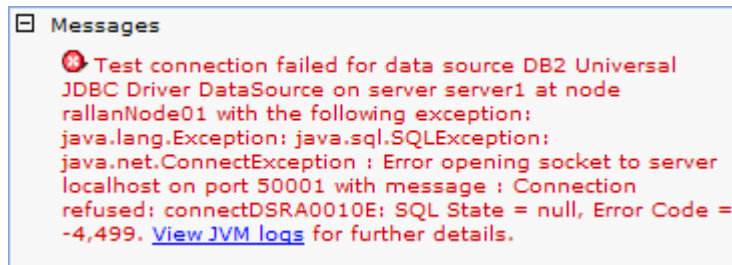
Details —

Additional information —

Transition statement —

Data source database parameter problems: Identification

- The data source is the connection between the application server and the database.
- Common configuration problems consist of:
 - Incorrectly specified database server parameters
 - Incorrect user authentication credentials
- Use the **Test Connection** service and examine the error message:



- As suggested by this error message, the JVM logs are a good place to begin the problem determination activity

Figure 10-11. Data source database parameter problems: Identification

WA5711.0

Notes:

The SystemErr.log holds the complete stack trace of the exceptions that occur when a data source connection test fails. The administrative console, provides various views in the *Troubleshooting* section of the left hand menu. Exceptions events can be viewed separately or, a portions of the log files can be viewed. Because the log files can become quite large, the administrative console includes a range filter for the number of lines that are displayed.

When inspecting the log files directly from the file system, it is important to remember that the log files from the node containing the failure will include the exceptions and trace information. For a JDBC connection, runtime exceptions are typically available from the log files of the deployment manager. However, if the JDBC provider scope is focused at the node or server level, it is necessary to inspect the logs on the node that raised the exception.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Data source database parameter problems: Diagnosis and resolution

```
3/8/06 11:13:22:458 MST] 0000000a SystemErr      R java.net.SocketException:  
Operation timed out: connect:could be due to invalid address at  
java.net.PlainSocketImpl.socketConnect(Native Method)
```

- The SystemErr.log tells you that the address provided was invalid for the connection.
- A connectivity problem could be as a result of an incorrect database name, server name, or port number.
 - If this is a large organization, the database administrator might have to be consulted to verify that database name.
 - You can verify that the server name is correct by attempting a ping of the machine or a Telnet session.
 - The port number could have been changed to something other than the default
 - If it is a DB2 database, the port number can be discovered through the DB2 Configuration Assistant or by exporting the instance parameters by using the DB2 Control Center.

Figure 10-12. Data source database parameter problems: Diagnosis and resolution

WA5711.0

Notes:

Determining the database name, server name, and port number will vary depending on the database vendor. In a development environment, it is common for the development team to administer the database themselves. Therefore, access to the machine and the configuration of the database will be something that the developers can discover for themselves. In a production environment, however, machine access is commonly controlled by dedicated administrators and even by separate groups. Therefore, it could be necessary to engage the database administrator in order to resolve the connectivity issue.

Instructor notes:

Purpose —

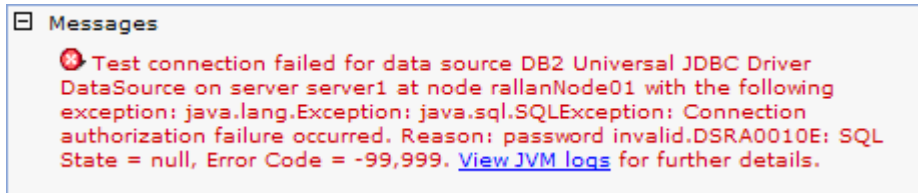
Details —

Additional information —

Transition statement —

Data source database authentication problems: Identification

- If you run the **Test Connection** service and receive an error message like:



- It is signifying a problem with the authentication credentials. This could be a result of:
 - Misspelled user name or password
 - Assigning the wrong JCA credential definition to the data source
 - Missing user account for this user on the database server machine

Figure 10-13. Data source database authentication problems: Identification

WA5711.0

Notes:

Prior to WebSphere Application Server Version 5, data source configuration also included the user name and password information. Beginning with WebSphere Application Server Version 5, J2EE Connector Architecture (JCA) was introduced, and the same set of credentials can be mapped to any number of data sources through the authentication widgets of the data source configuration panel. WebSphere Application Server Version 6.1 allows for the creation of a *Version 4 Data Source*. The connection test activity is the same as when using a JCA compatible data source configuration. If a credential failure occurs, then the credentials associated with the *Version 4 Data Source* will need to be verified. If you have a large number of these legacy data sources and the credentials are incorrect, then they will need to be fixed in each data source configuration. This is one of the advantages of using a JCA compliant authentication scheme: You only need to fix the one set of credentials and all the data sources will pick up the correction.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Data source database authentication problems: Diagnosis and Resolution

- The exception in the SystemErr.log file says that the password is invalid:

```
[3/23/06 10:40:16:492 CST] 0000003e SystemErr      R java.lang.Exception:  
java.sql.SQLException: Connection authorization failure occurred. Reason: password  
invalid.DSRA0010E: SQL State = null, Error Code = -99,999
```

- However, in this particular case, it was as a result of assigning the incorrect JCA authentication definition to the data source. Therefore, it is important to inspect all of the parameters associated with the user authentication and verify that they are correct.
- This might include coordination with the database administrator or server security administrator to verify that the user account actually exists and is a member of the group that is able to access the database.

Figure 10-14. Data source database authentication problems: Diagnosis and Resolution

WA5711.0

Notes:

Verify that user credentials include activities such as logging into the database server and verifying access to the machine. Depending on the database, security services could be provided by the local operating system, LDAP server, or some other form of access control. In a larger organization, verifying the user credentials could involve several people and cross over several groups. As was discussed previously, a simple Telnet test to the database server might be necessary in order to verify that the credentials are actually correct and have the necessary authorities required for the data source.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Overview of the DB2 Universal Driver trace facility

- Any industry strength database driver will have a trace or debug facility available.
- The DB2 Universal Driver has extensive trace capabilities that can be activated from the WebSphere Application Server V6.1 administrative console.
 - In addition to setting the trace level of the driver, the output file for the trace information can also be specified.
 - The amount of information that is logged depends on the trace level. A value of -1 will log everything available to the trace file.

Figure 10-15. Overview of the DB2 Universal Driver trace facility

WA5711.0

Notes:

The trace settings, output and interpretation will vary from vendor to vendor. The **Custom properties** panel for a given data source will be populated with parameters that are specific to the driver implementation of the vendor. For JDBC providers and data sources supported by WebSphere Application Server Version 6.1, custom properties will be available. However, for some user defined database connections, such as MySQL, no custom properties are automatically populated in the trace facility panel. In this case, you will need to refer to the documentation in order to activate driver tracing.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Enabling the DB2 Universal Driver trace facility

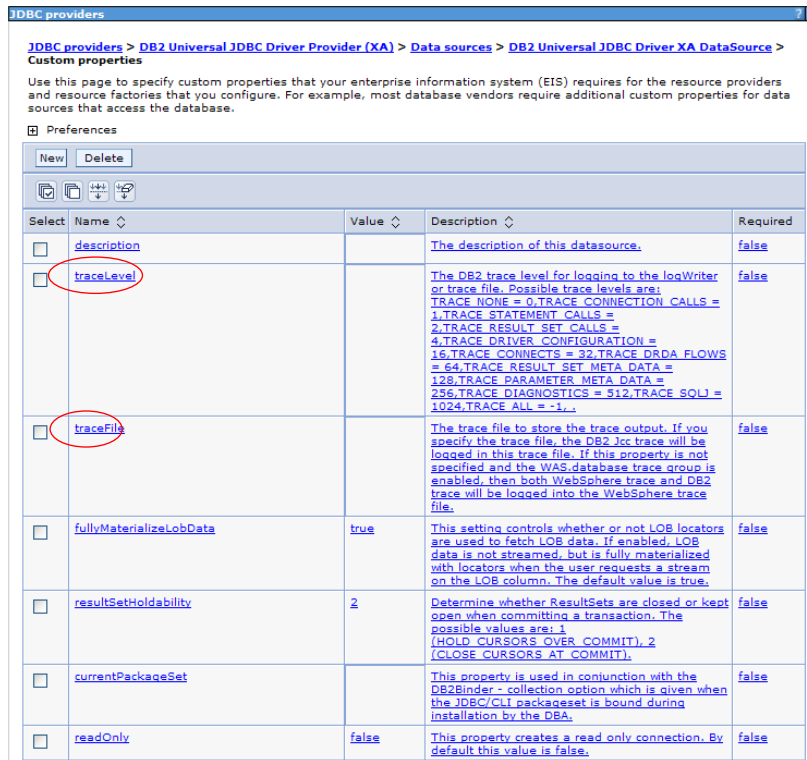


Figure 10-16. Enabling the DB2 Universal Driver trace facility

WA5711.0

Notes:

The DB2 Universal Driver Trace facility is located on the administrative console at **Resources -> JDBC -> JDBC Providers -> DB2 Universal JDBC Driver Provider -> Data sources -> Data source name -> Custom properties**.

When tracing DB2 drivers, the two parameters of interest are:

1. traceLevel – In this example, it has been set to -1, full trace.
2. traceFile – In this example, the trace file is called db2trace.log and, by default, is written to this file under the WAS_INSTALL_ROOT/(application server) directory because of the scope of the JDBC provider. Any relative path can be specified along with the name.

The trace file name is optional and, if not provided, the trace information will be logged into the WebSphere Application Server V6.1 trace file.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

DB2 Universal Driver trace facility output example

```
[ibm] [db2] [jcc] [Time:1143567094964] [Thread:WebContainer :
0] [DB2ConnectionPoolDataSource@59247092] getPooledConnection (UserName, <escaped>)
called
[ibm] [db2] [jcc] BEGIN TRACE_DRIVER_CONFIGURATION
[ibm] [db2] [jcc] Driver: IBM DB2 JDBC Universal Driver Architecture 2.4.19
[ibm] [db2] [jcc] Compatible JRE versions: { 1.3, 1.4 }
...
[ibm] [db2] [jcc] END TRACE_DRIVER_CONFIGURATION
[ibm] [db2] [jcc] BEGIN TRACE_CONNECTS
[ibm] [db2] [jcc] Attempting connection to localhost:50001/ENTDBx
[ibm] [db2] [jcc] Using properties: { cliSchema=null, clientProgramId=null,...
currentPackagePath=null, portNumber=50001, serverName=localhost,...
traceFile=db2trace.log, useCachedCursor=true, dataSourceName=null,
fullyMaterializeLobData=true, databaseName=ENTDBx, kerberosServerPrincipal=null,
jdbcCollection=NULLID, clientUser=null, traceLevel=-1, currentRefreshAge=-
...
[ibm] [db2] [jcc] END TRACE_CONNECTS
...
[ibm] [db2] [jcc] BEGIN TRACE_DIAGNOSTICS
[ibm] [db2] [jcc] [SQLException@65877092] java.sql.SQLException
[ibm] [db2] [jcc] [SQLException@65877092] SQL state = 08004
[ibm] [db2] [jcc] [SQLException@65877092] Error code = -4499
[ibm] [db2] [jcc] [SQLException@65877092] Message = The application server
rejected establishment of the connection. An attempt was made to access a
database, ENTDBx, which was not found.
[ibm] [db2] [jcc] [SQLException@65877092] Stack trace follows
com.ibm.db2.jcc.a.DisconnectException: The application server rejected
establishment of the connection. An attempt was made to access a database,
ENTDBx, which was not found.
    at com.ibm.db2.jcc.c.cb.u(cb.java:1595)
```

17

Figure 10-17. DB2 Universal Driver trace facility output example

WA5711.0

Notes:

The trace file can grow very large, extremely quickly. Attempting to debug a driver problem in a production environment could affect performance and result in a trace file that is difficult to view and transfer. Obviously there will be situation where only production systems will expose some class of problems. This is probably less true when resolving runtime reference issues, such as those associated with JDBC. Therefore, whenever possible, create a simple test that isolates the problem and execute it only once or a few times with tracing activated.

The *Test connection* facility uses a simple query: `SELECT 1 FROM TABLE1`. The trace file information in this example was generated using the *Test connection* facility and shows the driver information and the exception that is logged as a result of an improperly configured database name in the data source properties.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. True or False: The only JDBC driver implementations that are supported by WebSphere Application Server V6.1 are those that have pre-defined environment variables.
2. True or False: A JDBC Provider is analogous to a JCA connection factory.
3. True or False: The WebSphere Application variables that reference all driver implementations are pre-configured for the common JDBC providers.
4. Name two common sources of JDBC provider configuration error.

Figure 10-18. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1. T/F? Why?
2. T/F? Why?
3. T/F? Why?
4. Provide two answers:
 - a.
 - b.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

1. True or False: The only JDBC driver implementations that are supported by WebSphere Application Server V6.1 are those that have pre-defined environment variables.
 - **False. WebSphere Application Server V6.1 supports any JDBC compliant database. New variables can be defined in order to describe the driver implementation in the environment.**
2. True or False: A JDBC Provider is analogous to a JCA connection factory.
 - **False. The data source is analogous to a JCA connection factory.**
3. True or False: The WebSphere Application variables that reference all driver implementations are pre-configured for the common JDBC providers.
 - **False. Environment variables need to be configured for some driver implementations. Common drivers are pre-configured.**
4. Name two common sources of JDBC provider configuration error.
 - **Incorrectly specified environment variable value**
 - **Missing jar or library file**

Figure 10-19. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe the role of the JDBC provider
- Describe the role of the data source
- Understand some of the common problems encountered when configuring database connections
- Test a database connection using the WebSphere Application Server Version 6.1 *Test Connection* service
- Troubleshoot some of the common problems encountered when testing database connections

Figure 10-20. Unit summary

WA5711.0

Notes:

This concludes the unit on JDBC connection problem determination.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 5: Troubleshooting database configuration problems

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 11. Connection pool tuning and management problems

Estimated time

00:45

What this unit is about

This unit describes how to troubleshoot connection pool tuning and management problems.

What you should be able to do

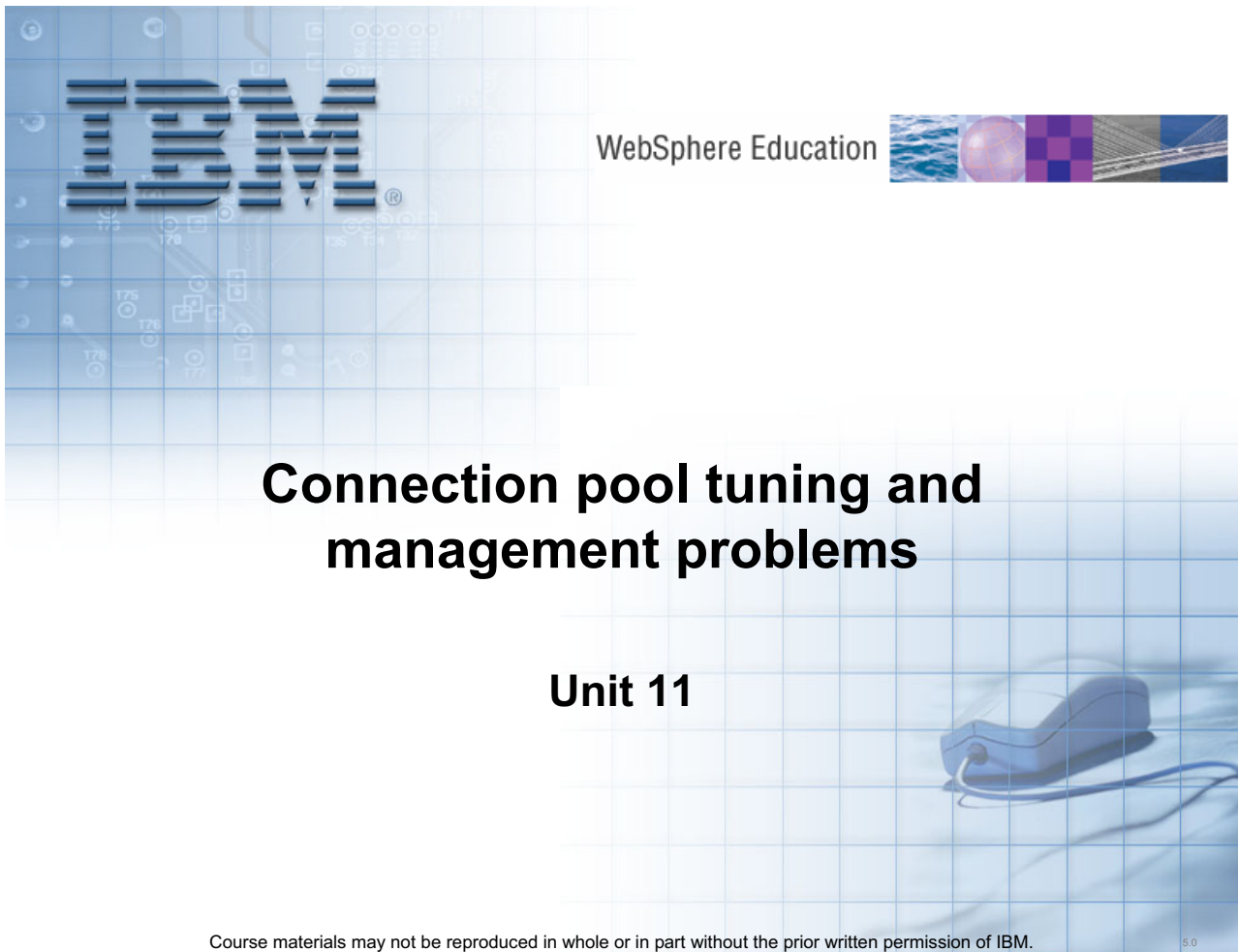
After completing this unit, you should be able to:

- Identify connection pool problems
- Describe what to look for in the WebSphere Application Server logs
- Enable tracing for connection manager components
- Interpret and analyze the trace data
- Explain how to use Tivoli Performance Viewer (TPV) to monitor a connection pool

How you will check your progress

Accountability:

- Machine exercises



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.0

Figure 11-1. Connection pool tuning and management problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Identify connection pool problems
- Use Tivoli Performance Viewer (TPV) to monitor a connection pool and generate tuning advice
- Enable tracing for connection pool manager components and interpret the trace data
- Perform problem determination tasks to find the root cause of a connection pool problem

Figure 11-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

What is connection pooling?

- Application server maintains a pool of ready-to-use connections to a data store
- Application client requests a connection using a data source or connection factory object
- Connection is retrieved from pool
- Benefits:
 - Minimizes database session setup and tear-down
 - Improves application database access performance
 - Spreads out connection cost over repeated uses

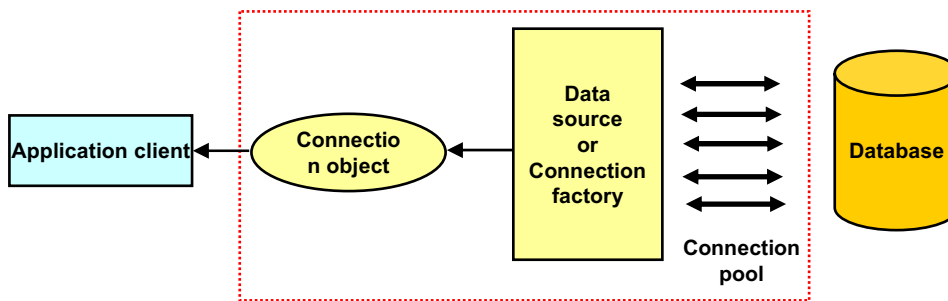


Figure 11-3. What is connection pooling?

WA5711.0

Notes:

Creating a connection to a database is an expensive operation.

- Connections are large objects (1-2 MB each) and take up system resources.
- Creating a connection from scratch is relatively time consuming.

The goal of connection pooling is to improve the efficiency and performance of database access by maintaining a pool of readily available persistent connections to the database.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

JCA connection pooling architecture

- WebSphere Application Server implements the J2EE Connector Architecture (JCA) V1.5 specification
- Connection pooling is supported by two components:
 - JCA connection manager (called J2C connection pool manager in WebSphere)
 - Relational resource adapter

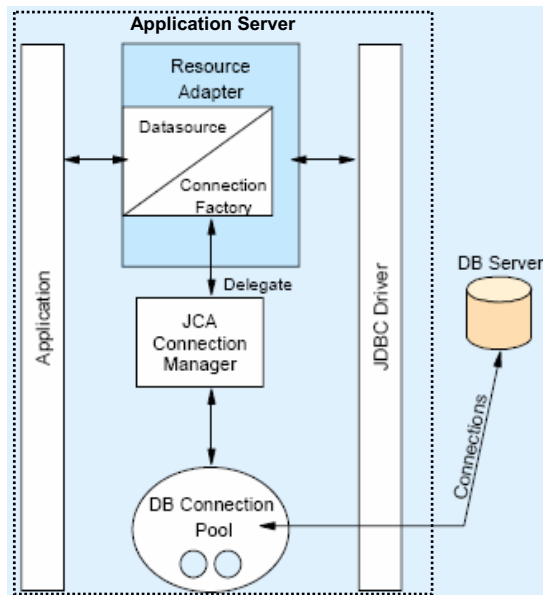


Figure 11-4. JCA connection pooling architecture

WA5711.0

Notes:

J2C = Java 2 Connector

JDBC = Java Database Connectivity

A resource adapter is a system-level software driver that a Java application uses to connect to an enterprise information system (EIS). A resource adapter plugs into an application server and provides connectivity between the EIS, the application server, and the enterprise application. WebSphere Application Server supports any resource adapter that implements version 1.0 or 1.5 of this specification.

The JCA connection manager provides connection pooling, local transaction support and security services. In WebSphere, it is implemented by the J2C connection pool manager. IBM supplies resource adapters for many enterprise systems separately from the WebSphere Application Server package.

WebSphere Application Server does provide the WebSphere Relational Resource Adapter implementation providing access to relational databases. This resource adapter provides data access through JDBC calls to access the database dynamically. The connection

management is based on the JCA connection management architecture and provides connection pooling, transaction, and security support. The WebSphere RRA is preinstalled and runs as part of WebSphere Application Server, and needs no further administration.

The relational resource adapter provides the JDBC wrapper or JCA Common Client Interface (CCI) implementation that allows applications to access the database JDBC driver.

The following steps describe how an application gets a connection from the pool:

- An application acquires a data source or connection factory object from the relational resource adapter.
- The data source or connection factory delegates the connection allocation request to the JCA connection manager.
- The JCA connection manager retrieves a free connection from the pool or creates a new one if none is available.
- The retrieved or created connection is returned to the application.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Types of connection pools in WebSphere

- JDBC provider connection pool
 - Enables access to a relational database
 - Provides a connection using a JDBC data source
 - Support for WebSphere Version 4 data sources also provided
 - Managed in the administrative console under JDBC Providers

- JMS provider connection pool
 - Enables access to the data store used by the default messaging engine or a WebSphere MQ provider
 - Provides a connection using a JMS connection factory
 - Managed in the administrative console under JMS Providers

- EIS Connection pool
 - Enables access to enterprise information systems such as CICS, legacy databases such as IMS and other back-end systems
 - Provides a connection using a data source or connection factory
 - Managed in the administrative console under Resource Adapters

- All are managed by the WebSphere **J2C connection pool manager**

Figure 11-5. Types of connection pools in WebSphere

WA5711.0

Notes:

JMS = Java Message Service

EIS = Enterprise Information Systems

CICS = Customer Information Control System

IMS = Information Management System

The J2C connection pool manager pools and manages all connections to a JCA compliant data store.

Note that support for WebSphere version 4 data sources is also provided for backwards compatibility, but uses the old WebSphere connection manager architecture. This is important because Version 4 data sources have their own problem areas and generate their own set of error messages.

The WebSphere V6 *default messaging engine* manages messaging resources and provides a pure JMS 1.1 implementation that is fully integrated with the application server process. It uses a JDBC database to store messages, transaction states and delivery records.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Sharable versus unshareable connections

- Shared connections
 - Single connection used by multiple getConnection calls within a same transactional context or container boundary
 - The connection is not released back to the pool even when closed
 - Connection is released when transaction or parent method completes
 - Default behavior – often more scalable

- Unshared connections
 - Each getConnection calls results in a connection being allocated from the connection pool
 - Connection is released back to the pool upon a close call
 - Multiple connections may be used within the same transaction

- Common issues
 - Shared connections may be held too long exhausting the pool and appearing as though a connection leak is present
 - Unshared connections may exhaust the pool as many may be used for each transaction

Figure 11-6. Sharable versus unshareable connections

WA5711.0

Notes:

Connection pooling in WebSphere Application Server is managed according to the Java Connector Architecture (JCA) and enables the use of shareable or unshareable connections. In WebSphere Application Server, connections are shareable by default. The use of shareable connections means that, if conditions allow it, different get connection requests by the application will actually receive a handle (indirectly) for the same physical connection to the resource. The benefits of this are improved performance and a reduction in the number of physical connections that need to be managed.

When the application closes a shareable connection it is not returned to the free pool. Rather, it remains ready for another request within the same transaction for a connection to the same resource. Shareable connections obtained by an application within a local transaction containment are kept reserved in the shared pool for use within that local transaction containment until the it ends, even if the application explicitly closes the connection. This behavior is beneficial to many applications, but can have unintended consequences for others.

The use of shareable connections can provide performance benefits in many situations, but you need to be aware of this default behavior and take it into account when developing applications. Otherwise, the application could experience problems under heavy workload.

Possible drawbacks of shared connections

- Sharing within a single component (such as an enterprise bean and its related Java objects) is not always supported. The current specification allows resource adapters the choice of only allowing one active connection handle at a time.
- Connection may be held long after the application may have called the close method

Possible drawbacks of unshared connections:

- Inefficient use of your connection resources. For example, if within a single transaction you get more than one connection then you use multiple physical connections when you use unshareable connections.
- Wasted connections. It is important not to keep the connection handle open (that is, your application does not call the **close()** method) any longer than it is needed. As long as an unshareable connection is open, the physical connection is unavailable to any other component, even if your application is not currently using that connection. Unlike a shareable connection, an unshareable connection is not closed at the end of a transaction or servlet call.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Detecting connection management related problems

- Look in the WebSphere SystemOut.log and SystemErr.log files for the following types of messages:

Message prefix or type	Message source
DSRA or CWWRA	<ul style="list-style-type: none"> • JCA resource adapter
CONM or CWWCM	<ul style="list-style-type: none"> • WebSphere Version 4 connection manager • Legacy connection manager used to support J2EE 1.2 applications
J2CA or CWWJC	<ul style="list-style-type: none"> • J2EE connector (J2C Connection pool manager) • Most recent JCA 1.5 compliant connection manager
WSCL or CWWSC	<ul style="list-style-type: none"> • WebSphere client (J2EE application client manager)
WTRN or CWWTR	<ul style="list-style-type: none"> • WebSphere transaction manager
SQLException or database error code	<ul style="list-style-type: none"> • Database manager

Figure 11-7. Detecting connection management related problems

WA5711.0

Notes:

JCA connection management problems can appear in the WebSphere logs as error messages with signatures shown in the table above.

WebSphere version 4 connection manager: WebSphere version 4.0 provided its own JDBC connection manager to handle connection pooling and JDBC access which was not based on JCA. This support is included with WebSphere Application Server V5 and V6 to provide support for J2EE 1.2 applications.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Typical connection pool runtime problem symptoms

- Sporadic failure to connect to an existing data source or connection factory:
 - Application has been working and connecting to an existing data source or connection factory
 - You start to experience poor user response time and see intermittent failures in getting a connection

- Next, look at the WebSphere logs to see if it additionally shows:
 - No specific exception
 - Probable cause: Improperly tuned connection pool settings
 - `ConnectionWaitTimeoutException`
 - Probable cause 1: Improperly tuned connection pool settings
 - Probable cause 2: Connection leak due to poor coding
 - `StaleConnectionException`
 - Probable cause: Stale connection

Figure 11-8. Typical connection pool runtime problem symptoms

WA5711.0

Notes:

Once configuration problems have been resolved you may begin to see sporadic failures during database access.

The main symptom that points to a possible connection pool problem is when the application has been working and successfully finding the data source or connection factory, but now sporadically fails to get a connection.

You can start to analyze the problem by looking at the WebSphere `SystemOut.log` and `SystemErr.log` for any additional exception messages. The root cause will likely be one of the following:

- Improperly tuned connection pool settings
- Connection leak
- Stale connection

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Connection pooling problem determination path

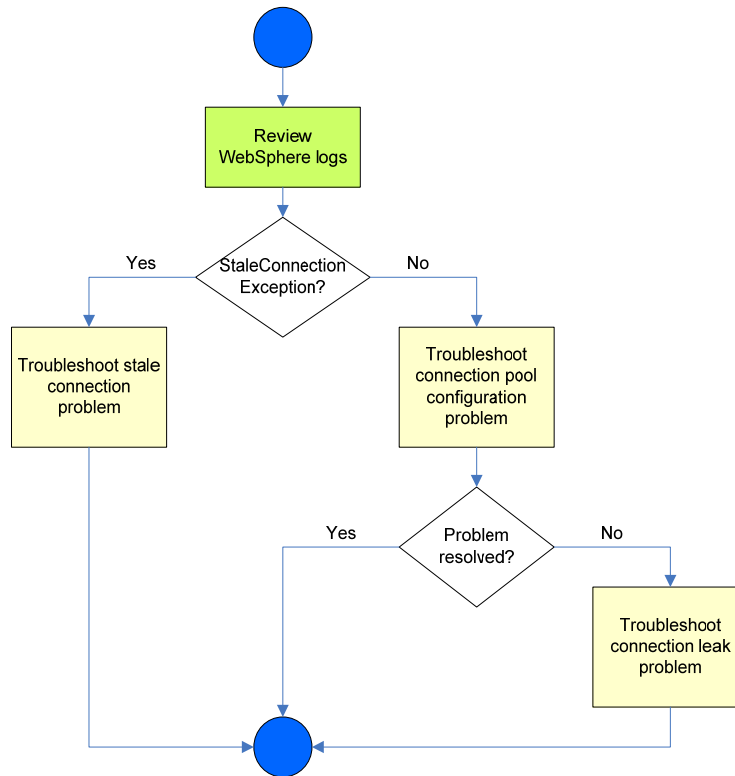


Figure 11-9. Connection pooling problem determination path

WA5711.0

Notes:

This diagram shows the recommended tasks and decision-based path to troubleshoot connection pooling problems. Keep in mind that prior to this point, you should have already confirmed that you have correctly configured connection parameters, and tested that you can properly establish a connection.

The next topics discuss each troubleshooting task in detail.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting connection pool configuration in the problem determination path

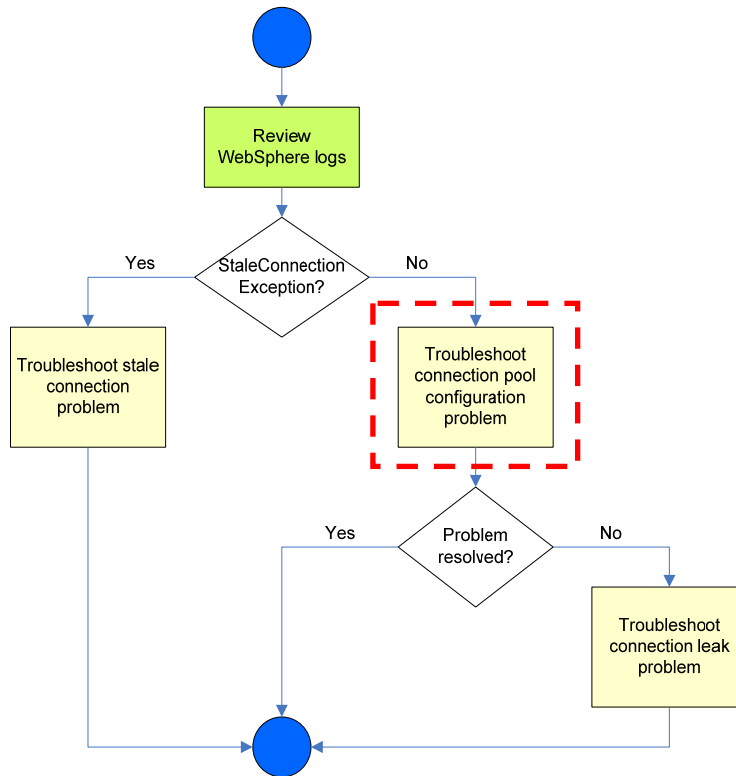


Figure 11-10. Troubleshooting connection pool configuration in the problem determination path

WA5711.0

Notes:

In the case where the symptom for a connection pool problem is not accompanied by a `StaleConnectionException` in the WebSphere logs, start your problem determination effort by looking at the connection pool configuration to rule out any performance tuning issues.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

The need for connection pool tuning

- Keeping connections in a pool consumes resources
 - Connections are large objects (1-2 MB each)
- An improperly tuned connection pool can result in:
 - Poor user response if the client is consistently waiting for a free connection
 - Application exceptions if the client cannot get a connection within the specified wait timeout interval
 - Reduced server throughput if unused connections are wasting system resources
- Connection pools need to be properly tuned to ensure optimal performance:
 - Maximize the chances that connections are available when needed
 - Minimize the number of idle connections
 - Minimize the number of orphaned connections

Figure 11-11. The need for connection pool tuning

WA5711.0

Notes:

Keep in mind each connection also ties-up resources on the database server itself. If multiple application servers are connected to the same database server and one application server is tying-up too many connections other application servers may start seeing problems too.

Sometimes executing queries take longer than expected and cause **getConnection()** requests to fail due to low connection timeout values or not enough connections in the pool. If pool parameters are set too small or too large, performance problems will result.

An orphaned connection is one that is allocated to a client but not being used and therefore abandoned, most likely because the client has experienced an unexpected problem.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Key connection pool parameters

- **Maximum connections**
 - Specifies the maximum number of connections that can be created in the pool
 - Default value is 10
 - A value of 0 allows the number of physical connections to grow infinitely and causes the Connection timeout value to be ignored

- **Connection timeout**
 - Specifies the interval, in seconds, after which a connection request times out and a `ConnectionWaitTimeoutException` is thrown.
 - Default value is 180 seconds (three minutes)
 - A value of 0 instructs the pool manager to wait as long as necessary until a connection becomes available

Figure 11-12. Key connection pool parameters

WA5711.0

Notes:

The pool contains physical connections to the back-end resource. When the maximum number is reached, no new connections are created, and the requester waits until one that is currently in use is returned to the pool. If this does not happen before the *Connection timeout* time interval, a *ConnectionWaitTimeoutException* is thrown.

The *Connection timeout* value indicates the number of seconds that a request for a connection waits when there are no connections available in the free pool and no new connections can be created because the *Maximum connections* value has been reached. If a connection is not available within this time, the pool manager throws a *ConnectionWaitTimeoutException*.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Connection pool parameters in the administrative console

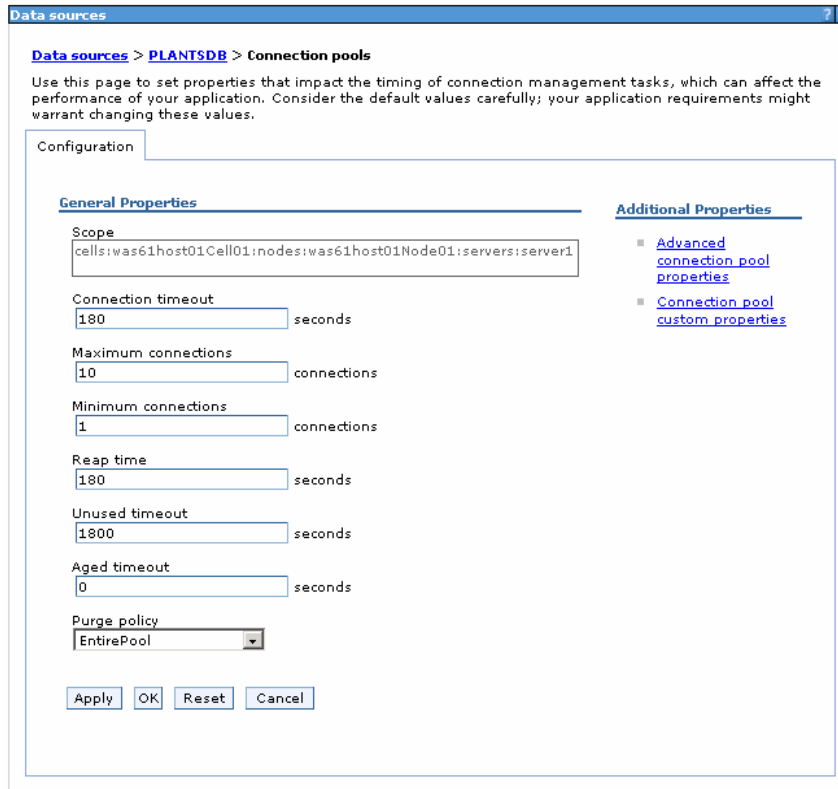


Figure 11-13. Connection pool parameters in the administrative console

WA5711.0

Notes:

The following additional connection pool properties can be configured in the administrative console:

Minimum connections: Specifies the minimum number of physical connections to be maintained. Until this number is reached, the pool maintenance thread does not discard any physical connections. However, no attempt is made to bring the number of connections up to this number. The default value is one connection.

For example, if *Minimum connections* is set to three, and one physical connection is created, that connection is not discarded by the *Unused timeout* thread. By the same token, the thread does not automatically create two additional connections to reach the *Minimum connections* setting.

Reap time: Specifies the interval, in seconds, between runs of the pool maintenance thread. The default value is 180 seconds.

For example, if *Reap time* is set to 60, the pool maintenance thread runs every 60 seconds. The *Reap time* interval affects the accuracy of the *Unused timeout* and *Aged timeout* settings. The smaller the interval you set, the greater the accuracy.

When the pool maintenance thread runs, it discards any connections that have been unused for longer than the time value specified in *Unused timeout*, until it reaches the number of connections specified in *Minimum connections*. It also discards any connection that remain active longer than the time value specified in *Aged timeout*.

Unused timeout: Specifies the interval in seconds after which an unused or idle connection is discarded. The default value is 1800 seconds.

For example, if the *Unused timeout* value is set to 120, and the pool maintenance thread is enabled (*Reap time* is not 0), any physical connection that remains unused for 120 seconds (two minutes) is discarded. Note that the accuracy of this timeout, as well as its performance, is affected by the *Reap time* value.

Aged timeout: Specifies the interval in seconds before a physical connection is discarded, regardless of recent usage activity. The default value is 0 which indicates that active connections are to remain in the pool indefinitely.

For example, if the *Aged timeout* value is set to 1200, and the *Reap time* value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that the accuracy of this timeout, as well as its performance, is affected by the *Reap time* value.

Purge policy: Specifies how to purge connections when a stale connection or fatal connection error is detected. Valid values are *EntirePool* (the default) and *FailingConnectionOnly*.

If you use *EntirePool*, all physical connections in the pool are destroyed when a stale connection is detected. Final destruction of connections which are in use at the time of the error might be delayed. However, these connections are never returned to the pool.

If you choose *FailingConnectionOnly*, the pool manager attempts to destroy only the stale connection.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Connection pool tuning tasks

- Monitor connection pool runtime behavior
 - Enable PMI and select desired statistic set
 - View connection pool performance metrics using Tivoli Performance Viewer (TPV)
 - Generate tuning advice using TPV Performance Advisor
- Tune connection pool parameters
 - Make one change at a time
 - Apply recommendations and best practices
- Test application
 - Use a load generation tool to simulate production-like loads
 - Compare results with original baseline
 - Document results
- Repeat Monitor-Tune-Test cycle until problem is resolved and performance goals are met

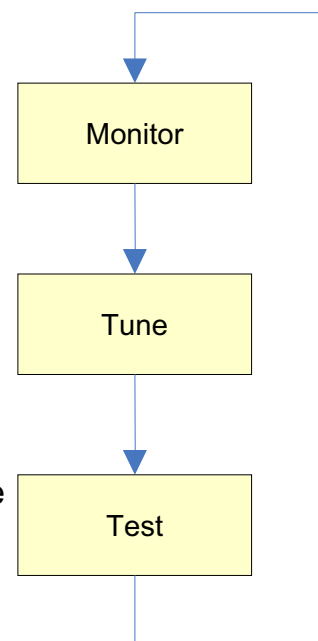


Figure 11-14. Connection pool tuning tasks

WA5711.0

Notes:

Performance tuning, in general, is an iterative and incremental process consisting of multiple Monitor-Tune-Test cycles. Having the right tools, including a load generation tool to simulate real-world users for load testing, is a must to ensure successful results.

The *art* of performance tuning is a mixture of documentation, test data, and experience. There are some tools that can assist with this practice such as the *Tivoli Performance Advisor* embedded in WebSphere Application Server V6, but the suggestions that it offers still need to be verified through load testing. The general method for getting the correct value is to *divide and conquer* by increasing the timeout and connection parameters until the timeout issue disappears and then backing them off until any wasted resources are recovered.

There is also a runtime performance advisor available known as the Performance and Diagnostic Advisor.

Note that the Performance Monitoring Infrastructure (PMI) is enabled by default in WebSphere V6.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — The next slides discuss these tasks in detail.

Monitoring the connection pool using TPV

- The following key connection pooling performance metrics should be monitored:

Metric Name	Description	What to look for
PoolSize	Size of the connection pool	<ul style="list-style-type: none"> •Increases as new connections are created (up to the value of <i>Maximum connections</i>) and decreases when connections are destroyed •A significant number of creates and destroys is an indication that the pool size (<i>Maximum connections</i>) should be adjusted •Counter is already enabled as part of the <i>Basic</i> (default) PMI statistic set
PercentUsed	Average percent of the pool that is in use	<ul style="list-style-type: none"> •If consistently low, you may want to decrease the pool size •Counter is already enabled as part of the <i>Basic</i> (default) PMI statistic set
WaitingThreadCount	Average number of threads that are concurrently waiting for a connection	<ul style="list-style-type: none"> •The optimal value for the pool size is that which reduces this value •Counter is already enabled as part of the <i>Basic</i> (default) PMI statistic set
PercentMaxed	Average percent of the time that all connections are in use	<ul style="list-style-type: none"> •Ensure that you are not consistently maxed a 100% •Counter requires the selection of either <i>All</i> or <i>Custom</i> PMI statistic set

Figure 11-15. Monitoring the connection pool using TPV

WA5711.0

Notes:

These metrics are collected by the PMI of WebSphere and available for display in TPV under the *JDBC Connection Pools* and *JCA Connection Pools* modules.

To access these modules, in the administrative console:

- Navigate to **Monitoring and Tuning -> Performance Viewer -> Current Activity**.
- Select the server you want to monitor.
- Expand **Performance Modules -> JDBC Connection Pools** or **JCA Connection Pools**.

At this point, to view the collected metrics for a JDBC data source:

- Expand the JDBC provider name for the data source.
- Check the check box for the data source. A chart appears showing graphs for selected data source counters.

For a complete list of the available PMI connection pool counters, refer to the WebSphere Information Center chapter on “J2C connection pool counters” at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/welcome_nd.html

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

TPV connection pool monitoring example

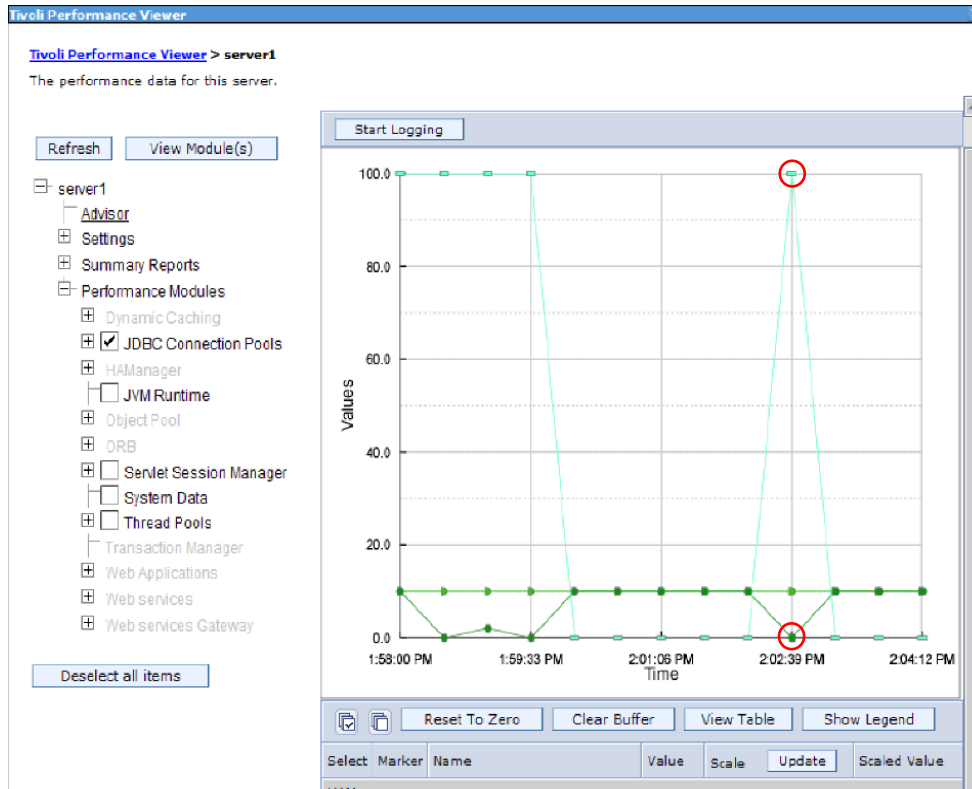


Figure 11-16. TPV connection pool monitoring example

WA5711.0

Notes:

This Tivoli Performance Viewer screen capture shows an example of connection pool graphs for a data source:

- The cyan colored graph plots the *PercentUsed* metric (average percent of the pool that is in use).
- The dark green graph plots the *FreePoolSize* metric (number of free connections in the pool).
- The green graph plots the *CreateCount* metric (total number of connections created). 10 connections have been created reaching the default *Maximum connections* value for the pool.

Notice that, as expected, when the *FreePoolSize* is 0 indicating no connection available in the pool, the *PercentUsed* value is at 100%.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Generating tuning advice using TPV Performance Advisor

- TPV Performance Advisor can provide configuration advice for connection pool size:
 - Advice appears in the Performance Advisor section of TPV
 - Based on collected PMI data over the last one minute interval
 - Uses IBM-defined rules of thumb for advice basis
- Limitations:
 - Pool sizing advice may not be generated if your timeout values are too high (pools are not returning back to minimum values)
 - TPV Advisor only gives recommendations when CPU usage is greater than or equal to 50%

Figure 11-17. Generating tuning advice using TPV Performance Advisor

WA5711.0

Notes:

WebSphere provides two separate tuning advisors:

- TPV Performance Advisor: Runs on demand and outputs advice to the TPV graphical user interface in the administrative console
- Performance and Diagnostic Advisor (Runtime Performance Advisor): Runs in the background and outputs advice to SystemOut.log and the administrative console under **Troubleshooting -> Runtime Messages -> Warning**.



Note

For the Performance and Diagnostic Advisor, it is possible to configure the **Minimum CPU For Working System** parameter to specific the threshold at which usage advice will be generated. This allows for a lower threshold of 50% (the default value).

The following examples illustrate the advice that TPV Performance Advisor would give in certain situations:

- Situation: The number of connections is low (equal to *Minimum connections*).
Advice: Decrease pool size.
- Situation: All data source connections are heavily used and heap space is available.
Advice: Increase maximum pool size.
- Situation: The size of the pool is fluctuating a lot (high variance), possibly indicating batch processing and wasted resources.
Advice: Decrease pool size.

To run the TPV Performance Advisor, in the administrative console:

- Navigate to **Monitoring and Tuning -> Performance Viewer -> Current Activity**.
- Select the server for which you want to generate advice.
- Select **Advisor** and look at the generated advice list.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Performance Advisor tuning advice example

[Runtime Events](#) > [Message Details](#)

Runtime events propagating from the server

General Properties

Message
 TUNE0206W: Decreasing the connection pool for data source jdbc/TradeDataSource by setting the minimum size to 0 and the maximum size to 3 may improve performance. Additional explanatory data follows. Pool utilization: 1.2%. This alert has been issued 1 time(s) in a row. The threshold will be updated to reduce the overhead of the analysis.

Message type
 Warning

Explanation
 Decreasing the size supports better pooling and frees memory resources.

User action
 From the administrative console, click: Resources > JDBC Providers > JDBC_provider > Data Sources > data_source > Connection pool properties.

Message Originator
 com.ibm.ws.performance.tuning.serverAlert.TraceResponse

Source object type
 RasLoggingService

Timestamp
 Oct 19, 2004 9:28:35 AM EDT

Thread Id
 15

Node name
 laptop-rzhouNode01

Server name
 server1

[Back](#)

Figure 11-18. Performance Advisor tuning advice example

WA5711.0

Notes:

This slide shows an example of a connection pool advice generated by the Runtime Performance Advisor.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tuning the connection pool

- Goal is to create a large enough pool that can handle a peak load but does not unnecessarily take up system resources
 - Unused connections during non-peak periods can be controlled with the Minimum connections parameter

- In order to successfully tune the connection pool, you need to know two pieces of information:
 - The requests per second that occur during a peak
 - How long the database takes to respond to each type of operation
 - SELECT, INSERT, UPDATE, and so on

- Key parameters to tune:
 - Maximum connections
 - Optimal value for pool size is that which reduces the value of concurrent waiters for a connection (WaitingThreadCount)

 - Connection timeout
 - Value should be based on a combination of how long the database operations take to complete and the number of concurrent waiters for a connection

Figure 11-19. Tuning the connection pool

WA5711.0

Notes:

Tuning the connection pool settings for optimal performance during peak load is an iterative activity. It is only through trial and error that the correct parameter values will be discovered. In particular, the two parameters that will have the greatest effect on correcting connection pool configuration errors are:

- Connection Timeout
- Maximum Connections

If the time taken to complete a database operation is greater than the amount of time a thread is willing to wait for a resource (Connection Timeout), then increasing the number of available connections only, is not the best approach. Conversely, if the connections are short-lived, then only increasing their number could lead to the application server being overloaded in other areas during a peak because the extra connections are unnecessarily consuming resources. Also, the number of idle connections during off peak periods should be weighed against the pool ramp up time when a peak occurs. By understanding the nature of these parameters and the nature of the database operations that will occur during

a peak load, an optimal configuration can be achieved leading to optimal performance with the lowest possible overhead.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Connection pool tuning best practices

- **Maximum connections setting**
 - As a first guess, try doubling the number of the maximum connections parameter
 - If this solves the problem, then start reducing the number of connections to determine where the failure or bottleneck threshold is
 - Better performance is generally achieved if this value is set lower than the value for the maximum size of the Web container thread pool

- **Connection timeout setting**
 - If a `ConnectionWaitTimeoutException` is found in the WebSphere logs:
 - Obtain the average database operations duration for the application
 - Start with a value that is 5 seconds longer than this average
 - Gradually increase it until problem is resolved or setting is at the highest value that the client will tolerate

- **Before you increase the pool size, consult the database administrator**
 - Ensure that the database server is configured to handle the maximum pool size setting
 - In a clustered environment, there is the potential of simultaneously allocating Max x N connections where:
 - Max = Maximum connections pool size
 - N = Number of clones

Figure 11-20. Connection pool tuning best practices

WA5711.0

Notes:

The database connection pool *Minimum* and *Maximum connections* values are often misunderstood. If you set a maximum of 40 connections and a minimum of 10 connections, the pool will not start with 10 connections created. The value of 10 connections minimum, is actually a low water mark. Until there are 10 connections required concurrently, the pool will only contain the maximum amount of concurrent connections required up to that point. Therefore, if the number of concurrent connections has only ever reached six, then the pool will contain 6 connections. Once the number of connections needed exceeds 10, the number of connections in the pool will not drop below 10 until the pool is cleaned out. In other words, after the reap time expires, all unused connections will be destroyed until the *Minimum connections* threshold is reached.

Configuration of the data sources should be done in consultation with the database administrator. For instance, the connection pool size should not be larger than the number of agents or connections allowed on the database server. This can become an issue, especially if cloning is used, because each application server will allocate its own pool. To compute the maximum connections the database may see, multiply the connection pool size by the size of the cluster. For example, assume the connection pool maximum is 10

(the default), and you have a deployment of 2 instances of an application server on each of 2 hosts. There is a potential for up to 40 connections to be open against the database simultaneously.

An application that does significantly more INSERT and UPDATE operations than SELECT operations will require greater resources at the database server. If auto-indexing is activated then the database could be spending a lot of its time re-indexing after each INSERT or UPDATE. If auto indexing is turned off, then any SELECT operations could become more expensive because the indexes have become stale. In either case, greater overhead will be incurred for applications that are modifying the data in the database.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting connection leaks in the problem determination path

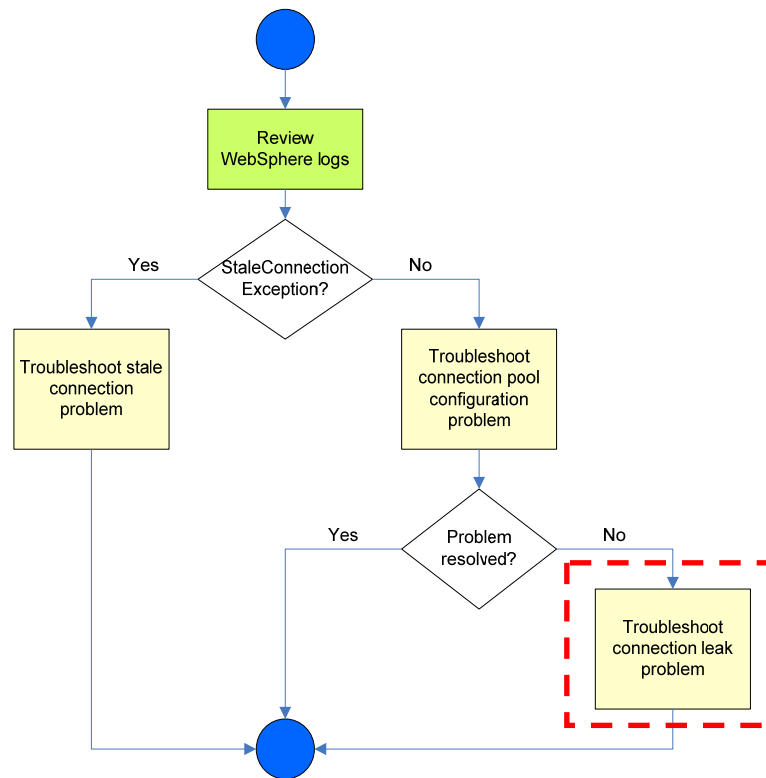


Figure 11-21. Troubleshooting connection leaks in the problem determination path

WA5711.0

Notes:

After you have ruled out a stale connection and connection pool tuning problem, consider the possibility of a connection leak.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Connection leaks

- A connection leak is a situation that arises when allocated connections are not properly released back to the pool after use.
- It causes:
 - User response time to increase
 - Eventual system lock-up if all worker threads are waiting for a connection
 - `ConnectionWaitTimeoutExceptions` to be thrown when the connection timeout threshold is reached

Figure 11-22. Connection leaks

WA5711.0

Notes:

A connection leak is typically identified by a `ConnectionWaitTimeoutException` in the WebSphere logs. WebSphere will eventually time-out orphaned connections and return them to the pool, but for an application that makes frequent use of database connections, this might not be enough. New connections can get queued up waiting for the database while old connections are waiting to be timed out. This can bring the application grinding to a halt, and you can see `ConnectionWaitTimeoutExceptions`.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Common causes of connection leaks

- Poorly-written applications often do not properly release database connections
 - Forget to call **connection.close()**
 - Appears most often in an exception case
 - Connections should be closed in a **finally{}** block
 - Also caused by one method getting a connection, invoking multiple methods, and then forgetting to close the connection when done

- Orphaned connections will only return to the pool after timeout
 - Can cause a back-up of new connections waiting for old connections to time-out
 - New connections that have waited too long throw a `ConnectionWaitTimeoutException`

Figure 11-23. Common causes of connection leaks

WA5711.0

Notes:

Applications can suffer from performance problems and even appear to hang if they do not close their connections properly. This is most often caused by developers not properly using the **connection.close()** method. To ensure that connections will be closed properly, they should be closed in a **finally{}** block.

Connection leaks have traditionally been hard to diagnose because the error messages do not usually provide specific enough information about the source of the problem. Usually a source code review is needed to find locations in the code where connections are not properly closed.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Connection leak diagnosis tasks

- Enable connection leak trace facility
 - Use administrative console

- Run and monitor application
 - Wait for `ConnectionWaitTimeoutExceptions` to occur

- Review and analyze trace file
 - Locate source of leak and resolve problem

Figure 11-24. Connection leak diagnosis tasks

WA5711.0

Notes:

The resolution of a connection leak problem requires the application developer to review the code and correct the problem. The primary solutions are:

- Review application to make sure that connections are properly closed
- Modify application to user fewer connections

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — The next slides discuss these tasks in detail.

Connection leak trace facility

- A connection leak trace facility is available in WebSphere to provide detailed diagnostic information
 - Prints stack traces of all open connections to trace.log when a `ConnectionWaitTimeoutException` occurs
 - Enables you to narrow the search for the responsible source code
 - Light-weight with lower performance overhead than standard connection manager tracing (1-5% impact)
- Limitation:
 - Connection leak trace facility only prints a stack trace of those connections that have been in use for more than 10 seconds

Figure 11-25. Connection leak trace facility

WA5711.0

Notes:

The connection pool manager has new instrumentation that can hold stack traces for all code that calls `getConnection()`.

When a thread times out waiting on a connection from a full connection pool, it will throw a `ConnectionWaitTimeoutException`. When this exception is thrown, the connection leak tracer will print out the stack traces for every open connection. It does so only when a problem has occurred, providing instant recognition of when it occurred and reduced overhead (1-5%) compared to the WebSphere tracing mechanism.

This feature is useful because it shows you the call stacks for all open connections at the time of the exception. This enables you to significantly narrow your search area when you look at the source code of the application to try and find the responsible code. It is also helpful to IBM Support, because it will help distinguish between application problems and WebSphere defects.

When you enable the connection leak trace facility, for every time interval (the default is 10 seconds), the WebSphere connection pool manager checks how long a connection has been in use and prints the stack trace to the trace.log file. Currently the default time interval

is unchangeable. If you have a need to change the default value, contact IBM Support to obtain an interim fix that allows you to add a custom property to the data source configuration.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Enabling the connection leak trace facility

- Enabled using a standard trace string `WAS.j2c=finest`

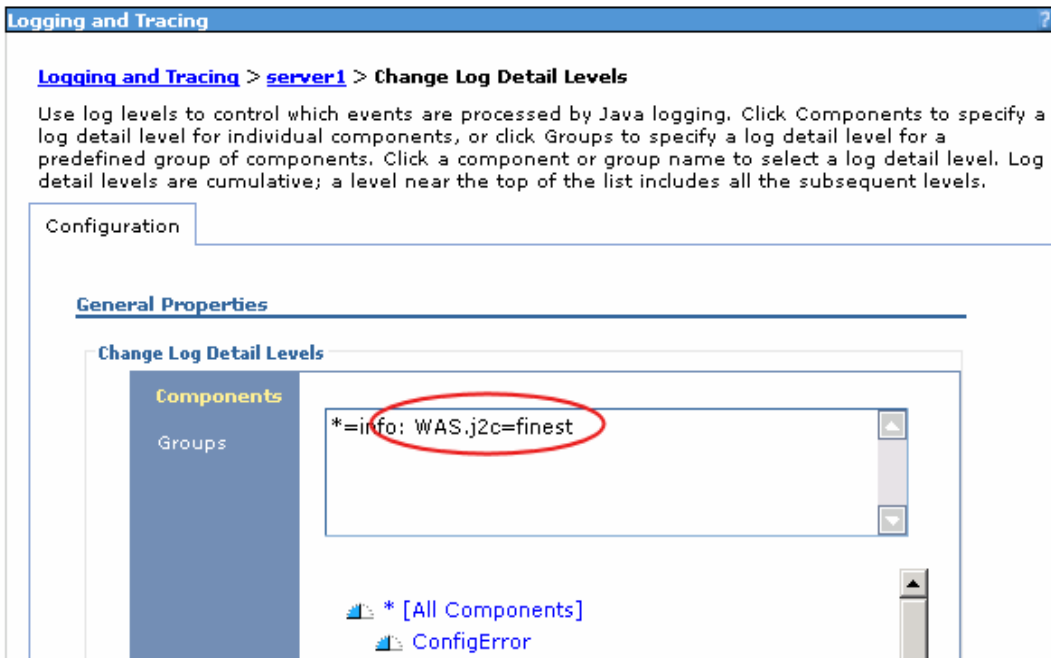


Figure 11-26. Enabling the connection leak trace facility

WA5711.0

Notes:

Connection leak diagnostics are enabled in the administrative console using the trace string shown above. To do so:

- Navigate to **Troubleshooting -> Logs and Trace**.
- Select the server where the application running.
- Select **Diagnostic Trace**.
- Make sure the **Enable Log** check box is checked.
- Select **Change Log Detail Levels**.
- Add the j2c trace string `WAS.j2c=finest` to the trace string text field.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

What to look for in the trace

- Search trace.log for the string: Connection Leak Logic Information
- If present, there are connections that have been in use for more than 10 seconds
- Analyze their stack trace to identify suspect application methods

```

Connection Leak Logic Information:
MSWrapper id df00df0 Managed connection WSRdbManagedConnectionImpl@a8c0a8c State:STATE_TI
Start time inuse Mon Jul 09 02:06:26 EDT 2007 Time inuse 11 (seconds)
Last allocation time Mon Jul 09 02:06:26 EDT 2007
getConnection stack trace information:
com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:712)
com.ibm.ws.rsadapter.jdbc.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:431)
com.ibm.ws.rsadapter.jdbc.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:400)
com.ibm.connleak.LeakServlet.doGet(LeakServlet.java:30)
javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
javax.servlet.http.HttpServlet.service(HttpServlet.java:856)
com.ibm.ws.webcontainer.servlet.ServletWrapper.service(ServletWrapper.java:989)
com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java:51)
com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest(ServletWrapper.java)
com.ibm.ws.webcontainer.webapp.WebApp.handleRequest(WebApp.java:3163)

```

Figure 11-27. What to look for in the trace

WA5711.0

Notes:

When a connection is leaked or held longer than the trace time interval (10 seconds), the trace contains:

- The string Connection Leak Logic Information
- Stack trace with all methods involved up to the **getConnection()** invocation

In the trace example above, the **doGet()** method of SnoopServlet.java is the suspected source of leaking connections.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting stale connections in the problem determination path

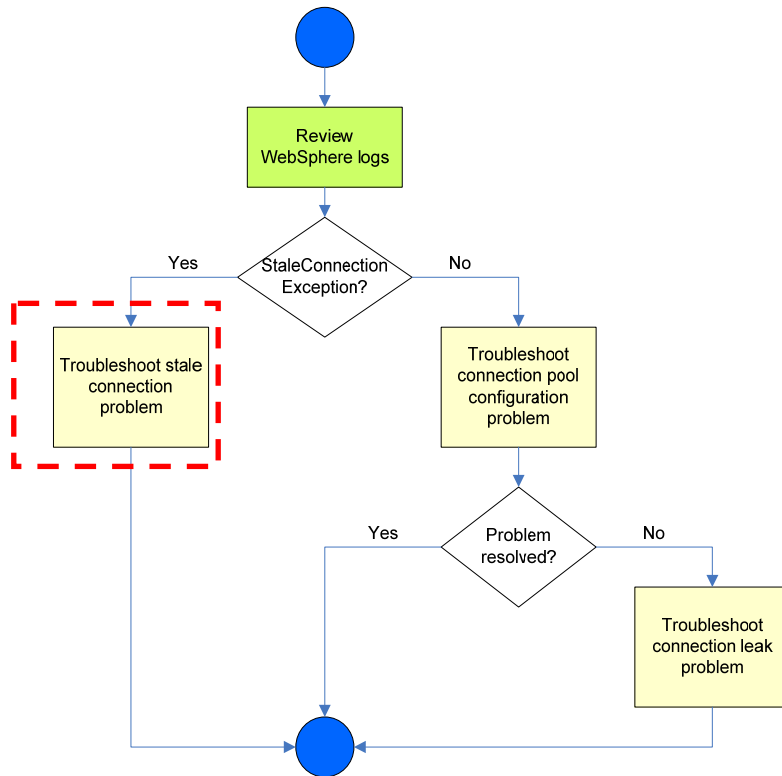


Figure 11-28. Troubleshooting stale connections in the problem determination path

WA5711.0

Notes:

When you receive the exception **com.ibm.websphere.ce.cm.StaleConnectionException**, follow the resolution steps for troubleshooting a stale connection problem.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Stale connections

- A stale connection problem arises when a connection held by a client is not longer valid.
- This situation can occur for many reasons, including the following:
 - A connection is no longer usable because of a database failure
 - An attempt is made to re-use an orphaned connection (applies only to version 4.0 data sources)
 - A connection is closed by the version 4.0 data source auto connection cleanup feature and is no longer usable

Figure 11-29. Stale connections

WA5711.0

Notes:

For a version 4.0 data source, a connection can orphaned because it has not being used it in at most two times the value of the *Unused timeout* setting. If the application tries to use it again, a stale connection condition occurs.

Auto connection cleanup is the standard mode in which connection management operates. This mode indicates that at the end of a transaction, the transaction manager closes all connections enlisted in that transaction. This enables the transaction manager to ensure that connections are not held for excessive periods of time and that the pool does not reach its maximum number of connections prematurely.

A negative ramification does ensue, however, when the transaction manager closes the connections and returns the connection to the free pool after a transaction ends. An application cannot obtain a connection in one transaction and try to use it in another transaction. If the application tries this, a *StaleConnection* exception occurs because the connection is already closed.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Recovering from a stale connection

- In general, a stale connection condition indicates that the connection to the database has gone bad
 - Connection cannot be recovered and must be completely closed rather than returned to the pool
- Recovering from stale connections is a joint effort between the application server run time and the application developer:
 - The application server will purge the connection pool based on its PurgePolicy setting and eliminate the bad connection
 - The application developer can explicitly catch a stale connection exception and programmatically recover from bad connections (for example, get a new one)

Notes:

Explicitly catching a *StaleConnection* exception while running within the context of a transaction will not cause the transaction to roll-back and provides the opportunity to programmatically recover from it, for example, by getting a new connection.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Other stale connection troubleshooting tasks

- Check database or firewall timeout settings
 - Consult with database administrator or network system administrator for the presence of these timeouts
 - If present, they can close connections and cause *StaleConnectionExceptions*

- Determine if a specific query is getting the exception
 - Examine the *SQLState* and *SQLCode* returned with the exception

- Trace the problem using one or more of the following options:
 - *WAS.database*
 - *RRA*
 - *WAS.j2c*

Figure 11-31. Other stale connection troubleshooting tasks

WA5711.0

Notes:

You might need to disable your database or firewall timeouts or review the connection pool settings so that they are suited to your environment. For example, the connection pool aged timeout should be less than the firewall timeout, which should be less than the database timeout.

If you are getting a *StaleConnectionException* when executing a certain query, it is likely that the query causes the JDBC driver to return the *SQLException* with an *SQLState* and *SQLCode* that are mapped to *StaleConnectionException*. Looking at these returned values can help you determine the root cause of the problem.

Sometimes a connection is unusable after an *SQLException* occurs, but the application server does not throw the *StaleConnectionException* because the *SQLState* and *SQLCode* are not the ones from the mapping list for *StaleConnectionException*. In this case, you should try to recreate the problem with a simple test case and trace it with the *WAS.database*, *RRA* and *WAS.j2c* trace options enabled. The trace should help you identify the query that is causing the problem and the *SQLException* that is returned by the JDBC driver.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Identify connection pool problems
- Use Tivoli Performance Viewer (TPV) to monitor a connection pool and generate tuning advice
- Enable tracing for connection pool manager components and interpret the trace data
- Perform problem determination tasks to find the root cause of a connection pool problem

Figure 11-32. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 6: Troubleshoot a connection pool problem

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 12. Security configuration-related problems

Estimated time

01:00

What this unit is about

This module describes how to detect and troubleshoot security-related problems.

What you should be able to do

After completing this unit, you should be able to:

- Describe common problems with WebSphere security
- Recognize symptoms of common security-related problems
- Analyze relevant log files for security messages
- Enable server tracing on relevant security components
- Analyze and interpret trace information
- Locate the security configuration files
- Use tools to validate the security configuration files

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

SG24-6451-00, *WebSphere Application Server V6 System Management and Configuration Handbook*, Ch. 2 “WebSphere Application Server v6 Architecture”, Section 2.11 “Security”

WebSphere Application Server V6 information center:
<http://publib.boulder.ibm.com/info center/wasinfo/v6r0/index.jsp>, Section: “Troubleshooting Security Configurations”

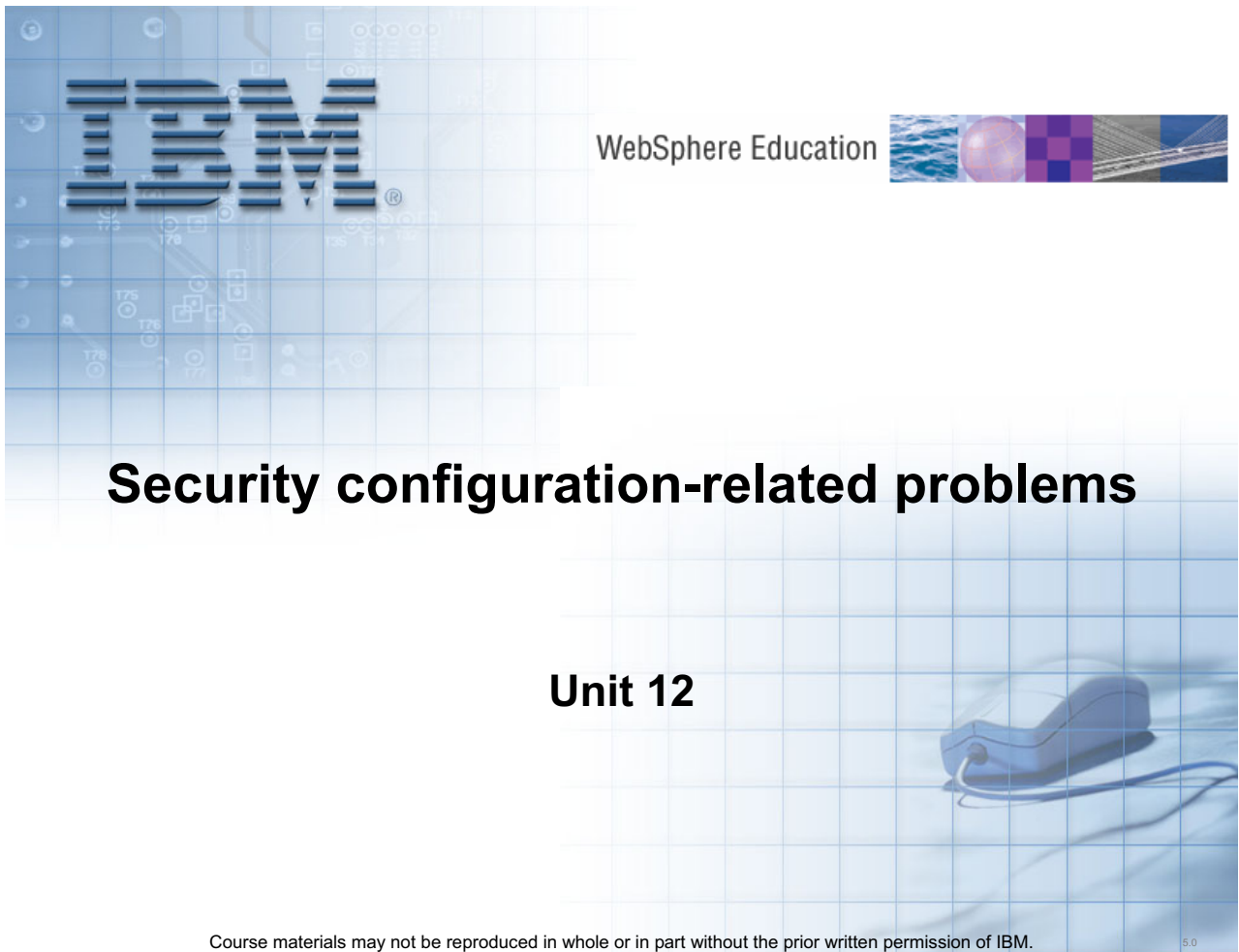


Figure 12-1. Security configuration-related problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe common problems with WebSphere security
- Recognize symptoms of common security-related problems
- Analyze relevant log files for security messages
- Enable server tracing on relevant security components
- Analyze and interpret trace information
- Locate the security configuration files
- Use tools to validate the security configuration files

Figure 12-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Review of security components and security flows

After completing this topic, you should be able to:

- List the WebSphere security components
- Describe authentication and authorization flows
- Describe virtual member manager architecture

Figure 12-3. Review of security components and security flows

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Security components overview

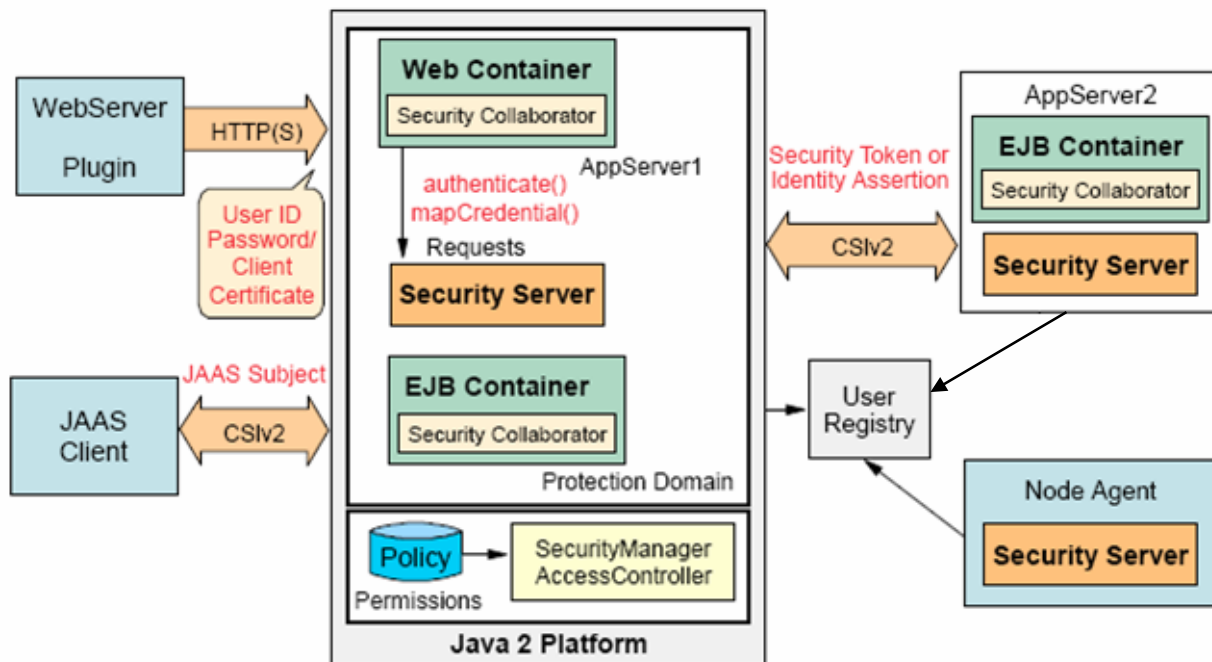


Figure 12-4. Security components overview

WA5711.0

Notes:

Security server

The security server is a component of WebSphere Application Server that runs in each application server process. If multiple application server instances are executed on a single node, then multiple security servers exist on that node.

The security server component is responsible for managing authentication and for collaborating with the authorization engine and the user registry.

Security collaborators

A security collaborator is an application server component that enforces security constraints specified by the deployment descriptors. The collaborator communicates with the security server every time authentication and authorization actions are required. The following security collaborators are identified:

The **Web security collaborator** resides in the Web container and provides the following services to the application:

- Checks authentication

- Performs authorization according to the constraint specified in the deployment descriptor
- Logs security tracing information

The **EJB security collaborator** resides in the EJB container. The EJB security collaborator uses Common Secure Interoperability Version 2 (CSIv2) and Secure Authentication Service (SAS) to authenticate Java client requests to enterprise beans. The EJB security collaborator works with the security server to perform the following functions:

- Checks authorizations according to the specified security constraint
- Supports communication with local user registry
- Logs security tracing information

Instructor notes:

Purpose — Review security components, protocols, and component interactions.

Details — Point out that security is distributed in an ND Cell in that each application server and node agent has its own security server. Note that the deployment manager is not shown in this diagram, but it also has its own security server. Point out the shared, remote User Registry as a possible single point of failure. Point out that both security collaborators log tracing information.

Additional information —

Transition statement —

Security flows: Web browser communication

- When a Web browser sends a request to a WebSphere application, the following security interactions occur:
 1. A Web user requests a Web resource that is protected by the application server
 2. The Web server plug-in receives the request and recognizes that the resource is on the application server
 3. Web server plug-in redirects the request to the Web container security collaborator which calls the JAAS login configuration if necessary
 4. If authentication is successful, the Web request reaches the Web container
 5. The Web security collaborator passes the following to the security server for authorization:
 - User credentials
 - Security information contained in the deployment descriptor from EAR file
 6. If the Web application subsequently calls an EJB
 - User credentials are extracted from the established security context
 - Authorization is performed by the EJB collaborator

Figure 12-5. Security flows: Web browser communication

WA5711.0

Notes:

Regarding the third point above, the Web authenticator does not necessarily perform authentication. If there is an LTPA token present then authentication is bypassed. If there is a client certificate present, then authentication happened at the Web server, and there is only an identity assertion to the Web container. If authentication does happen, the Web authenticator actually calls the JAAS login configuration and runs all the login modules in that configuration. The result of the login modules determines if the user is authenticated or not.

Instructor notes:

Purpose — Describe the process of authentication and authorization for a Web client attempting to access a secured resource.

The instructor may wish to refer the students to the diagram on the next slide.

Details —

Additional information —

Transition statement —

Authentication flows

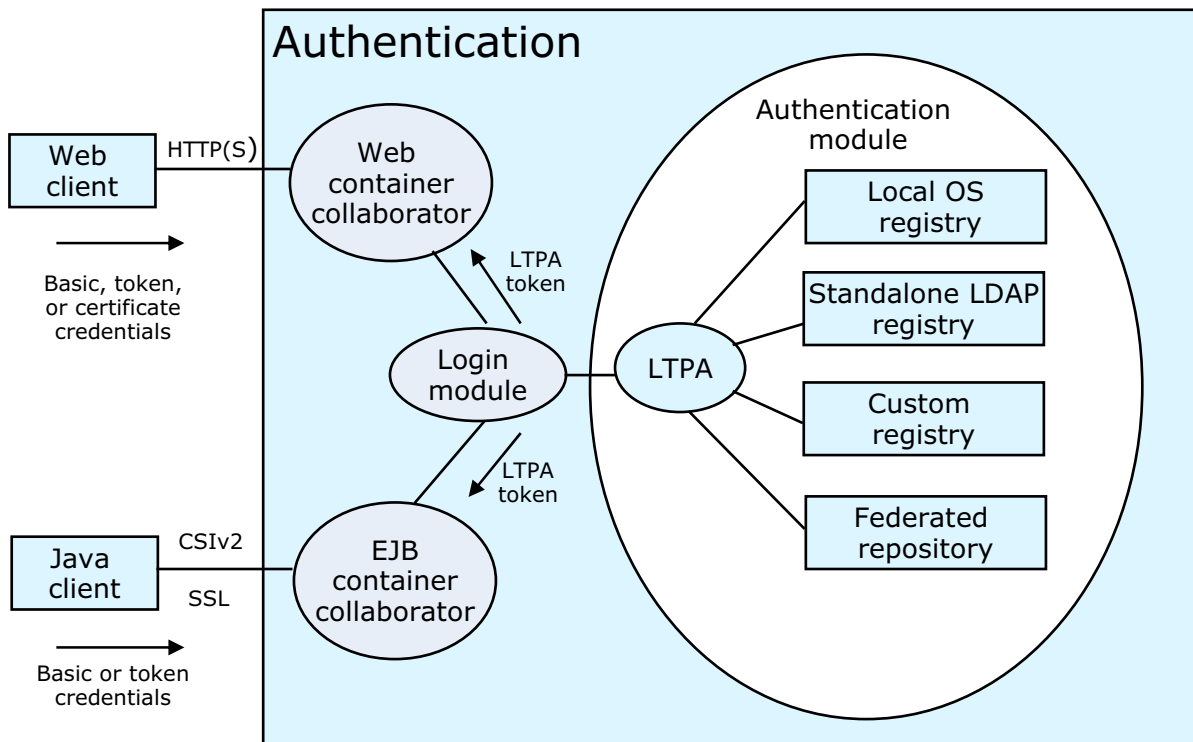


Figure 12-6. Authentication flows

WA5711.0

Notes:

The Simple WebSphere Authentication Mechanism (SWAM) is deprecated in WebSphere Application Server V6.1

Instructor notes:

Purpose — Describe the process of authentication and authorization for a Web client attempting to access a secured resource.

Use this diagram to illustrate the authentication process for a Web client and a Java client.

Details —

Additional information —

Transition statement —

Authorization flows

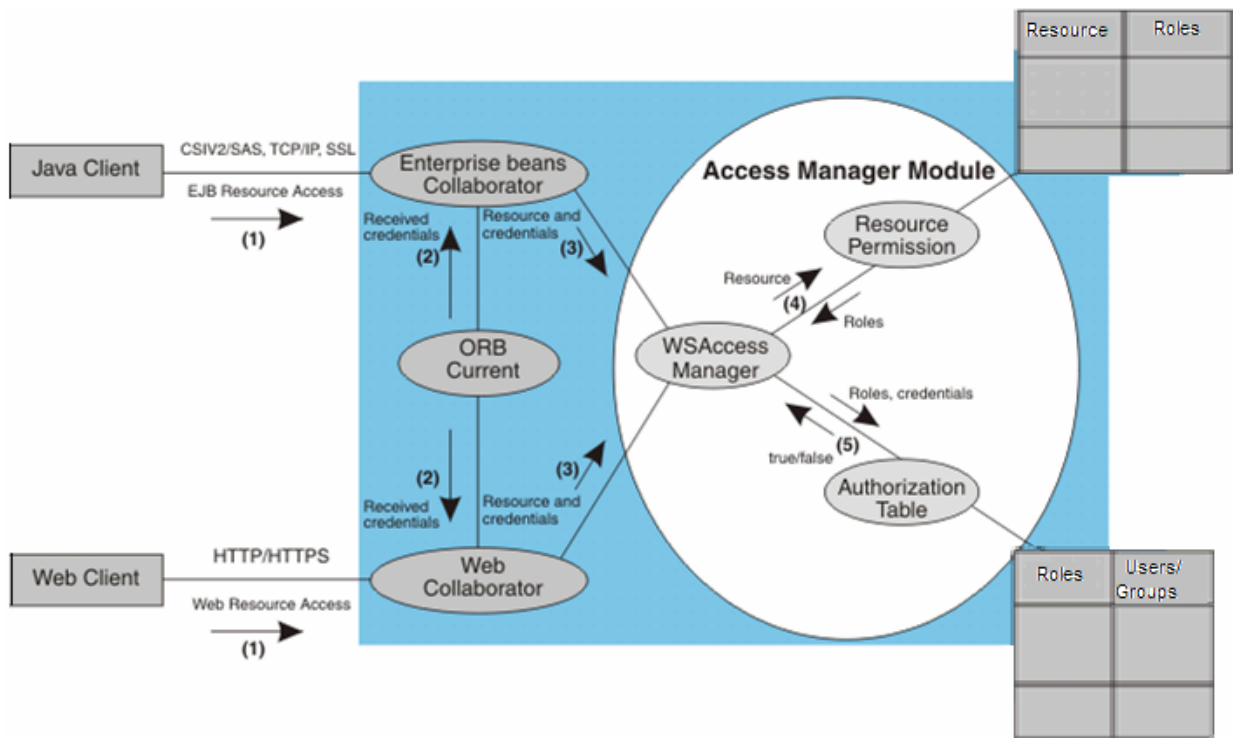


Figure 12-7. Authorization flows

WA5711.0

Notes:

Instructor notes:

Purpose — Use this diagram to illustrate the authorization process for a Web client and a Java client. Point out that authorization failures for particular uses depends on the security roles and the mapping of the user to those roles.

Details —

Additional information —

Transition statement —

Security flows: Java client communication

- When a Java client interacts with a WebSphere application, the following occurs:
 1. A Java client performs a JAAS login prior to making a business request
 2. The CSIv2 or IBM SAS interceptor performs authentication on the server side on behalf of the ORB, and sets the security context
 3. Subsequently the client makes a business request that reaches the server side ORB
 4. The server side ORB passes the request to the EJB container
 5. If the request is for a protected EJB method, the EJB container passes the request to the EJB collaborator
 6. The EJB collaborator reads the deployment descriptor from the EAR file and reads the user credentials from the security context
 7. Credentials and security information are passed to the security server, which validates user access rights and passes this information back to the collaborator
 8. After receiving a response from the security server, the EJB collaborator authorizes the client or denies access

Figure 12-8. Security flows: Java client communication

WA5711.0

Notes:

There is a basic difference between Web and EJB client authentication. For Web clients, authentication is deferred until the user attempts to access a protected resource, than the authentication process is initiated by the server. On the other hand EJB clients will explicitly perform a JAAS login independent of (and prior to) making a business request. So authentication and authorization for an EJB client will occur on separate calls.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Federated repositories

- Federated repositories provide:
 - Federation capabilities using the virtual member manager (VMM) component
 - Administrative utilities to manage existing users and groups through
 - The administrative console
 - Command line utilities
 - Public APIs
- Integrated with WebSphere Application Server security
 - As a user registry option
 - Federated Repositories option
- Ability to use multiple repositories simultaneously for user registry
 - File based (default out-of-box security)
 - Multiple LDAP directories
 - Databases

Figure 12-9. Federated repositories

WA5711.0

Notes:

The primary purpose of virtual member manager is to allow the management of user identities, profiles, and relationships. These management features are available through the administrative console, command line utilities, and public APIs. Additionally, the user management capabilities are also integrated into WebSphere Application Server security in several important ways. First, VMM provides a new user repository option called “federated repositories.” VMM also provides the appropriate JAAS and JACC framework to allow for application security using the federated repositories option.



Note

When WebSphere Application Server is installed the VMM component is automatically installed.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Federated user repositories

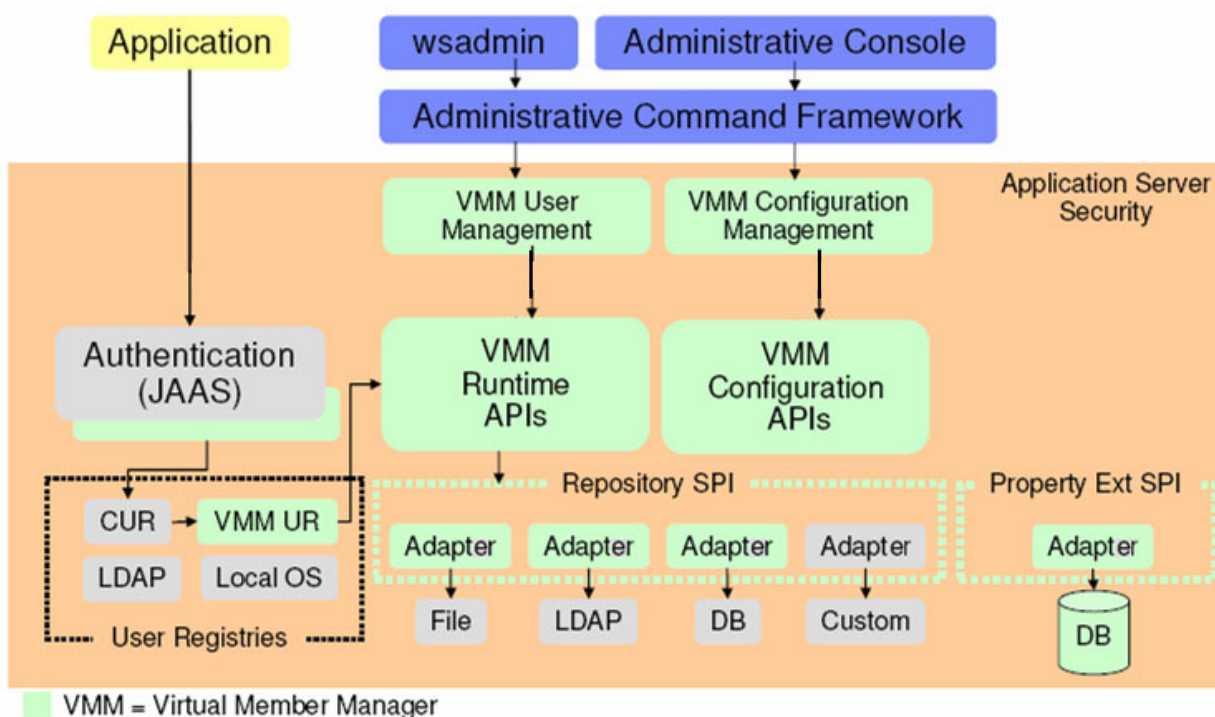


Figure 12-10. Federated user repositories

WA5711.0

Notes:

The diagram above illustrates how the virtual member manager component fits into the overall application server security architecture. Application requests interact with the JAAS framework for authentication. The authentication framework interacts with the user registries which can include Local OS, LDAP, CUR or a federated repository. The virtual member manager runtime APIs can access data stored in repositories by using adapters provided by the repository API.

The administrative actions taken either through wsadmin or the Administrative Console work through the same administrative command framework that uses the virtual member manager APIs.

The Repository SPIs (System Programming Interface) are defined in the interface:
com.ibm.wsspi.wim.Repository

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Common problems and troubleshooting methods

After completing this topic, you should be able to:

- Describe what can go wrong
- Describe how to approach a security problem
- Recognize normal security messages
- Determine if a problem is related to authentication or authorization
- Determine if a problem is SSL-related
- Examine a stack trace in the system log
- Enable tracing of security components
- Analyze trace information
- Troubleshoot SSL problems
- Troubleshoot Java 2 security problems

Figure 12-11. Common problems and troubleshooting methods

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

What can go wrong—The short list

- Errors trying to enable administrative security
 - Invalid user IDs
 - Problems accessing the user registry
- Errors after security is enabled
 - Authentication failures
 - Authorization errors accessing a Web page
- Access and login problems after security is enabled
 - Problems trying to log in to the Administrative Console
 - Access exceptions if applications are not prepared for Java 2 security
- Errors with the SSL configuration
 - Problems accessing resources with HTTPS URLs
 - SSL handshake exceptions
- Single signon configuration problems
 - Authentication failures, mismatching LTPA keys
- User authorization issues
 - Problems with user/group role mappings
- Server fails to start
 - Remote user registry inaccessible
 - Node synchronization problems

Figure 12-12. What can go wrong-The short list

WA5711.0

Notes:

Many security-related problems like the ones listed on this slide are documented in detail in the WebSphere V6.1 information center in the “Troubleshooting security configurations” section.

Instructor notes:

Purpose — Present a list of common security-related problems to begin the discussion of Security troubleshooting. Ask students to share some of their experiences with security troubleshooting.

Details —

Additional information —

Transition statement —

Approach to troubleshooting security-related issues

- **Does the problem occur when security is disabled?**
 - A good test to determine that a problem is security-related
 - Just because a problem only occurs when security is enabled does not always make it a security problem
 - More troubleshooting is necessary to ensure the problem is really security-related
 - Does the problem go away when Java 2 security is disabled?

- **Did security seem to initialize properly?**
 - A lot of security code executes during server initialization
 - Examine the SystemOut.log and SystemErr.log files to check for warnings and exceptions that are security-related

Figure 12-13. Approach to troubleshooting security-related issues

WA5711.0

Notes:

Instructor notes:

Purpose — A high-level approach involves confirm that it is a problem related to security, and if so confirm that security services started without any errors.

Details —

Additional information —

Transition statement —

Normal security initialization messages

- Messages generated in the SystemOut.log indicate normal code initialization of an application server

```

AuditServiceI A   SECJ6004I: Security Auditing is disabled.
distSecurityC I   SECJ0309I: Java 2 Security is disabled.
Configuration A   SECJ0215I: Successfully set JAAS login provider...
distSecurityC I   SECJ0212I: WCCM JAAS configuration information...
distSecurityC I   SECJ0240I: Security service initialization completed...
SASRas           A   JSAS0006I: Security connection interceptor initialized.
SASRas           A   JSAS0001I: Security configuration initialized.
SASRas           A   JSAS0003I: Authentication mechanism: LTPA
SASRas           A   JSAS0004I: Principal name:defaultWIMFileBasedRealm/
SASRas           A   JSAS0007I: Client request interceptor registered.
SASRas           A   JSAS0008I: Server request interceptor registered.
SecurityCompo A   JSAS0009I: IOR interceptor registered.
UserRegistryI A   SECJ0136I: Custom Registry... has been initialized
distSecurityC I   SECJ0243I: Security service started successfully
distSecurityC I   SECJ0210I: Security enabled true

```

Figure 12-14. Normal security initialization messages

WA5711.0

Notes:

The sequence of messages that are generated in the SystemOut.log indicate normal code initialization of an application server.

Note that in an ND Cell, the SystemOut.log for all the servers (Dmgr, node agents, application servers) should have security related entries identical to those shown above.

Note the message: JSAS0004I: Principal name:defaultWIMFileBasedRealm/

In WebSphere Application Server V6.1, this message provides the name of the security realm, not the server principal. In this case you see the default realm name for federated repositories. If you were using a stand-alone LDAP registry the message would be something like: JSAS0004I: Principal name:<ldapserver>:389/

Prior to V6.1, the actual server principal name was shown in this message.

Instructor notes:

Purpose — Emphasize that administrators should know what log messages reflect a “normal” startup on a particular server. In addition to the success messages above, there may be other warning messages or even error messages that might be “normal” for a particular server in that they have been noticed in the past but the server runs OK otherwise. Sometimes such warning/error messages are explained in the Release Notes.

Details —

Additional information —

Transition statement —

Authentication or authorization problem

- Many security problems fall under one of these two categories

- Authentication is the process of determining who the caller is
 - When authentication fails, this is typically because:
 - Authentication data is not what was expected (wrong ID, wrong password)
 - User being authenticated is not in the registry or password is no longer valid
 - The registry is misconfigured or not accessible

- Authorization is the process of validating that the caller has the proper authority to invoke the requested method
 - When authorization fails, this is usually related to:
 - Application bindings from assembly and deployment
 - Identity of the caller who is accessing the method
 - Roles that are required by the method

Figure 12-15. Authentication or authorization problem

WA5711.0

Notes:

Instructor notes:

Purpose — Distinguish between authentication and authorization problems pointing out what components or configurations are most relevant in each case.

Details —

Additional information —

Transition statement —

Problem related to Secure Sockets Layer (SSL)

- SSL is a distinct layer of security
 - Problems are usually separate from authentication and authorization

- SSL problems are usually first-time setup problems because the configuration can be difficult

- Handshake exceptions are common
 - Each client must contain a valid signer certificate for the server
 - During mutual authentication, each server must contain valid client certificates

Figure 12-16. Problem related to Secure Sockets Layer (SSL)

WA5711.0

Notes:

IKeyman is the IBM key management GUI tool that is used to configure SSL. It can also be a valuable tool for troubleshooting SSL problems. This tool is launched from your application server or deployment manager's bin directory.

As of WebSphere Application Server V6.1, most of the IKeyman functionality is incorporated into the administrative console. The IKeyman tool is still available as a stand-alone application.

Instructor notes:

Purpose — Point out that configuring SSL over various network segments is a somewhat difficult administrative setup task.

SSL can be configured from the Web clients to the Web server, from the Web servers to the application servers, and from application servers to LDAP servers. Knowledgeable management of encryption keys and digital certificates is required in order to perform these configurations correctly.

Details —

Additional information —

Transition statement —

Stack trace in the system log file

- A single stack trace tells a lot about the problem
 - What code initiated the code that failed
 - What component is failing
 - Which class the failure actually came from

- Sometimes the stack trace is all that is needed to solve the problem
 - It may pinpoint the root cause

- Other times, it only gives a clue, and may be misleading

- When product support analyzes a stack trace and it is not clear what the problem is
 - They may request that you gather additional trace data
 - You may need to trace several security components with different levels of detail

Figure 12-17. Stack trace in the system log file

WA5711.0

Notes:

Instructor notes:

Purpose — As in most troubleshooting situations, the system log files should be viewed first for exceptions, errors, and, warning messages. Emphasize what info can be found, and what to look for, in the system log files.

Details —

Additional information —

Transition statement —

Example: SystemOut.log stack trace

- Symptom: The deployment manager appears to start successfully. However an examination of the SystemOut.log file of Dmgr shows many exceptions and stack traces. The beginning of the first stack trace looks like:

```
[7/2/07 15:46:03:849 EDT] 0000000a LdapRegistryI E SECJ0352E: Could
not get the users matching the pattern wsbind because of the following
exception javax.naming.CommunicationException: DM01:389 [Root exception
is java.net.ConnectException: Connection refused: connect]

    at com.sun.jndi.ldap.Connection.<init>(Connection.java:222)
    at com.sun.jndi.ldap.LdapClient.<init>(LdapClient.java:133)
```

Figure 12-18. Example: SystemOut.log stack trace

WA5711.0

Notes:

DM01 is the host name of the remote LDAP server.

Instructor notes:

Purpose — Have students try to figure out what this message is telling them. Why is the host unknown? What steps should be taken: Check that DM01 is running? Ping DM01 from the deployment manager machine?

Details — The message above was created by disconnecting the deployment manager machine from the network. Since the LDAP Server DM01 is remote, the deployment manager cannot access the user registry. Reconnecting the deployment manager machine to the network will, in this case, enable it start up without any errors.

Additional information —

Transition statement —

Example: SystemOut.log exceptions

- An attempt to log into the Administrative Console fails
 - The following error messages are found in the SystemOut.log

```
LdapRegistryI E   SECJ0336E: Authentication failed for user
wasadmin because of the following exception
com.ibm.websphere.security.CustomRegistryException:
java.net.ConnectException: Connection refused: connect

LTPAServerObj E   SECJ0369E: Authentication failed when using
LTPA. The exception is javax.naming.CommunicationException: DM01:389
[Root exception is java.net.ConnectException: Connection refused:
connect].

FormLoginExte E   SECJ0118E: Authentication error during
authentication for user wasadmin
```

Figure 12-19. Example: SystemOut.log exceptions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tracing security components

- Classes that implement WebSphere security:
 - **com.ibm.ws.security.***
 - com.ibm.ws.security.audit.*
 - com.ibm.ws.security.auth.*
 - com.ibm.ws.security.core.*
 - com.ibm.ws.security.ejb.*
 - com.ibm.ws.security.policy.*
 - com.ibm.ws.security.registry.*
 - com.ibm.ws.security.role.*
 - com.ibm.ws.security.util.*
 - com.ibm.ws.security.web.*
 - **com.ibm.websphere.security.***
 - com.ibm.websphere.security.WSSecurityHelper
 - com.ibm.websphere.security.WebSphereSecurityPermission
 - **SASRas**
 - **VMM**
 - com.ibm.ws.wim.*
 - com.ibm.websphere.wim.*
 - com.ibm.wsspi.wim.*

Figure 12-20. Tracing security components

WA5711.0

Notes:

Instructor notes:

Purpose — Point out the different security components that can be traced: auth, ejb, registry, and so forth. The use of wildcards (*) in trace specifications makes it easy to enable tracing on multiple components.

Details —

Additional information — Typically the customer should not turn on tracing unless directed to do so by IBM Support.

Transition statement —

MustGather tracing requirements

- For specific security problems, the MustGather documents require that certain components be traced.
 - Global security problems
 - `com.ibm.ws.security.*=all`
 - `SASRas=all:com.ibm.ws.security.*=all:ORBRas=all` (when EBJ authentication is involved)
 - Java 2 security problems
 - `*=info:com.ibm.ws.security.policy.*=all:com.ibm.ws.security.core.SecurityManager=all`
 - Federated repository problems
 - `com.ibm.ws.security.*=all:com.ibm.websphere.wim.*=all:com.ibm.wsspi.wim.*=all:com.ibm.ws.wim.*=all`
- Typically you should increase the **Maximum Number of Historical Files** from 1 to 10 in the Diagnostic Trace Service

Figure 12-21. MustGather tracing requirements

WA5711.0

Notes:

The MustGather documentation provides trace requirements for the following additional security problems:

- JAAS Web login problems
- Problems running as non-root
- Problems with SPNEGO

Used to implement Windows desktop single signon, SPNEGO stands for Simple and Protected GSSAPI Negotiation Mechanism. SPNEGO is a standard GSSAPI pseudo-mechanism for peers to determine which GSSAPI mechanisms are shared, select one and then establish a security context with it.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Result from security components trace

- Portion of trace.log file

```
LdapRegistryI > getUniqueId Entry wsbind
LdapRegistryI > getUsers Entry wsbind
LdapRegistryI > search Entry
LdapRegistryI 3   DN: dc=ibm,dc=com
LdapRegistryI 3   Search scope: 2
LdapRegistryI 3   Filter: (&(uid=wsbind)(objectclass=inetOrgPerson))
LdapRegistryI 3   Time limit: 3
LdapRegistryI 3   Attr[0]: 1.1
LdapRegistryI > getDirContext Entry
LdapRegistryI 3   try connect to ldap://DM01:389
LdapRegistryI 3   enterJNDI:P=231764:O=0:CT
LdapRegistryI 3   exitJNDI:P=231764:O=0:CT
LdapRegistryI 3   javax.naming.CommunicationException: DM01:389 [Root
exception is java.net.ConnectException: Connection refused: connect]
LdapRegistryI A   SECJ0418I: Cannot connect to the LDAP server
ldap://DM01:389.
                at java.net.PlainSocketImpl.socketConnect(Native Method)
```

The log shows the events leading up to the Communication Exception.

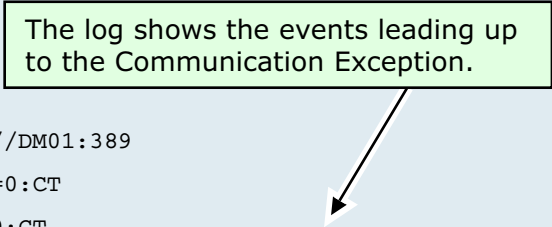


Figure 12-22. Result from security components trace

WA5711.0

Notes:

By default, when tracing is enabled for a server, the trace information is written to the trace.log file in the logs directory of the server.

Instructor notes:

Purpose — Point out that without tracing enabled at the detail level set on a previous slide, only the UnknownHostException info was visible in the systemOut.log with none of the preceding event details.

Details —

Additional information —

Transition statement —

SSL problems and Java 2 security problems

After completing this topic, you should be able to:

- Describe how SSL is used in WebSphere
- Identify the client/server SSL connection flows in WebSphere
- Recognize common problems that can occur during SSL handshakes
- Perform problem determination steps to diagnose and fix SSL related problems
- Identify the symptoms of Java 2 security problems
- Resolve Java 2 security access exceptions using the PolicyTool to edit the was.policy file of an application

Figure 12-23. SSL problems and Java 2 security problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SSL use in WebSphere

- SSL used by WebSphere to provide data encryption and authentication between a client and server
 - This includes connections to resources outside of WebSphere like LDAP and databases
- WebSphere uses JSSE as the SSL implementation that is provided by the IBM JRE
 - JSSE handles the SSL handshake and protection provided by SSL
- SSL can be configured between many different points in WebSphere
 - Browser to Web server
 - Plug-in to the application servers
 - Application servers to LDAP servers
 - Application servers to databases

Figure 12-24. SSL use in WebSphere

WA5711.0

Notes:

The Java Secure Socket Extension (JSSE) is a set of packages that enable secure Internet communications. It implements a Java technology version of Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols. It includes functionality for data encryption, server authentication, message integrity, and optional client authentication.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

How does SSL work?—The “handshake”

- SSL uses a combination of asymmetric and symmetric encryption to create a session between the client and server.
 - Asymmetric encryption is used to negotiate a session key (shared secret)
 - Asymmetric encryption is slow but does not require a shared secret
 - Symmetric encryption is used to transfer data between the client and server
 - Symmetric encryption is fast but requires a shared secret

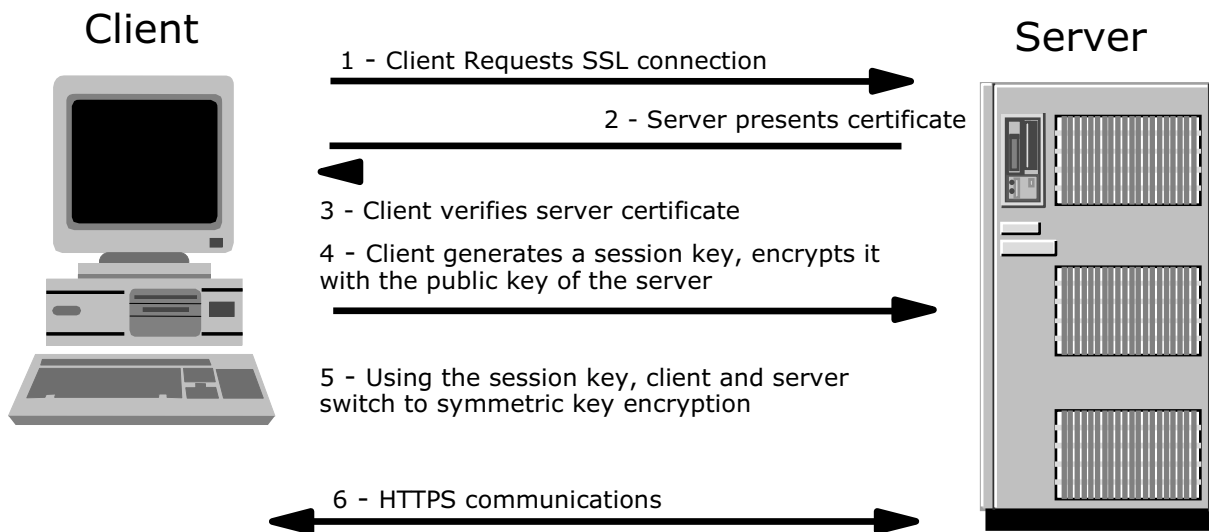


Figure 12-25. How does SSL work?—The “handshake”

WA5711.0

Notes:

Because the client chooses its own session key, nobody else knows it. It can securely send that session key to the server using the public key of the server. Now, nobody but the client and server know the session key. The session key is then used as a “shared secret” to switch to the much more efficient symmetric key encryption.

A certificate (or signing certificate) contains information about the server, including the public key of the server, and is digitally signed by the Certificate Authority.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SSL client/server identification

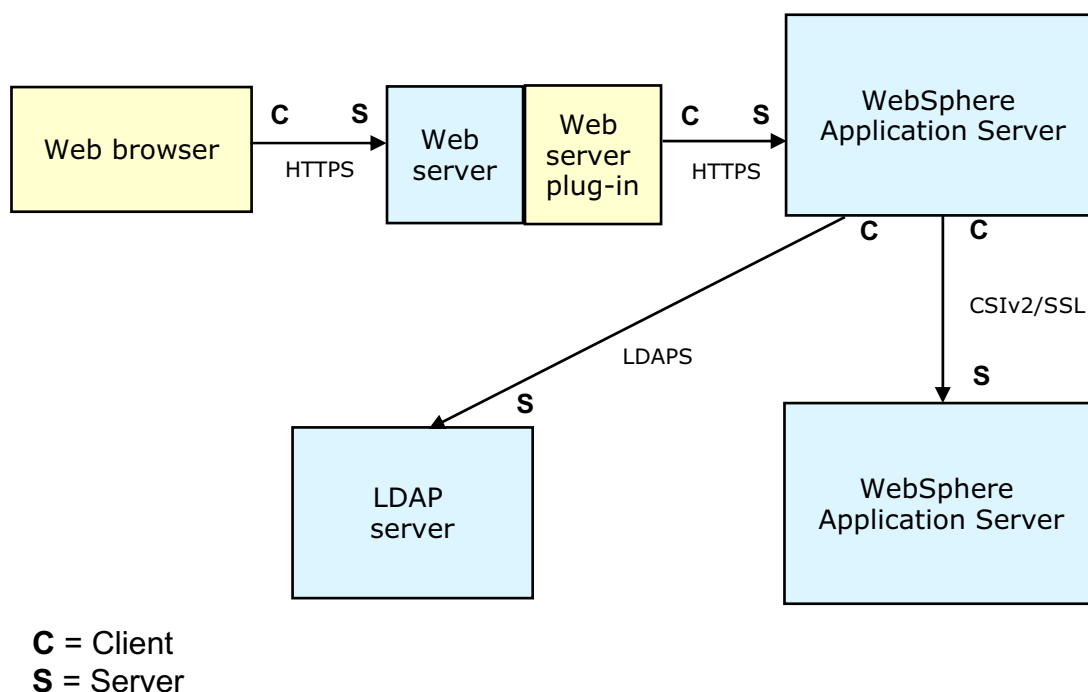


Figure 12-26. SSL client/server identification

WA5711.0

Notes:

This diagram illustrates the various client/server points in SSL communication between the user at the browser and the various points within WebSphere. Understanding who is the client and server during in an SSL connection allows the administrator to properly configure the keystores/truststores or other configuration.

Understand that “client,” in this context, may refer to a server process. It is the initiator of an SSL connection, so the Web server plug-in is a client to the application server, and one application server may be a client to another application server. Also, during mutual authentication, the server must contain the *signer* certificate of the client. Remember, that SSL mutual authentication is merely a statement of trust. It does not look at distinguished names or any other cert content, just verifies that the certificate was signed by a trusted signer. This is different to what WebSphere will do for an end-user certificate used for authentication, where it will verify that the user is actually in the registry.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SSL problems – Handshake failures (1 of 3)

- **javax.net.ssl.SSLHandshakeException: unknown certificate**
 - Some possible causes are:
 - Not having the public key for the target server in the client truststore file
 - The application is setting the following System Properties with WebSphere Security enabled
 - javax.net.ssl.keystore
 - javax.net.ssl.truststore
 - To correct these problems:
 - Export the public key of the server from the keystore file and import it as a signer certificate into the client truststore
 - The recommended solution is to use socket factories, which define their own keystore/truststore, without using the System properties.
 - For an example, go to:
 - <http://www.ibm.com/developerworks/java/library/j-customssl>
 - If socket factories cannot be used, the following technote provides methods to resolve this issue
 - <http://www.ibm.com/support/docview.wss?rs=180&uid=swg21191941>

Figure 12-27. SSL problems – Handshake failures (1 of 3)

WA5711.0

Notes:

A **key store** contains the personal certificates that can be used as the identity for the SSL end point referencing the key store. If more than one certificate is present, a certificate alias on the SSL configuration specifies one of the personal certificates. When an SSL connection is made (on either the client or the server side), certificates may be exchanged. The personal certificate referenced by the SSL configuration and stored in the key store is the certificate that will be used.

A **personal certificate** represents the identity of the end point and contains a public and private key for signing/encrypting data.

A **trust store** contains the signer certificates which this end point trusts when either making connections (from an outbound end point) or accepting connections (for an inbound end point).

A **signer certificate** represents a certificate and public key associated with some personal certificate. The purpose of the signer certificate is to verify personal certificates. By accepting the signer certificate into the trust store of an end point, you are allowing the owner of the private key to establish connections with this end point; that is, the signer

certificate explicitly trusts connections made to or by the owner of the associated personal certificate. The signer certificate is typically made completely public by the owner of the personal certificate, but it is up to the receiving entity to determine if it is a trusted signer prior to adding it to the trust store.

Socket factories are a simple way to capture a variety of policies related to the sockets being constructed, producing such sockets in a way which does not require special configuration of the code which asks for the sockets. Due to polymorphism of both factories and sockets, different kinds of sockets can be used by the same application code just by passing it different kinds of factories.

Factories can themselves be customized with parameters used in socket construction. So for example, factories could be customized to return sockets with different networking timeouts or security parameters already configured. The sockets returned to the application can be subclasses of `java.net.Socket`, so that they can directly expose new APIs for features such as compression, security, record marking, statistics collection, or firewall tunneling.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SSL problems – Handshake failures (2 of 3)

- **javax.net.ssl.SSLHandshakeException** - The client and server could not negotiate the desired level of security.
Reason: handshake failure
 - Some possible causes are:
 - Not having common ciphers between the client and server.
 - Not specifying the correct protocol.
 - To correct these problems:
 - Review the SSL settings in the administrative console
 - Check the property that is specified in the Protocol box matches the client/server
 - Check the cipher suites that are in the Selected Ciphers text box.
 - May need to add more cipher suites to the list
 - Correct the protocol or cipher problem by using a different client or server protocol and cipher selection
 - Typical protocols are SSL, SSLv3, TLS

Figure 12-28. SSL problems – Handshake failures (2 of 3)

WA5711.0

Notes:

The **ssl.client.props** file is located in

`<WAS_HOME>\profiles\<Profile_Name>\properties` contains global SSL properties.

Verify the following lines in this file.

```
com.ibm.ssl.protocol=
```

```
com.ibm.ssl.enabledCipherSuites=
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SSL problems – Handshake failures (3 of 3)

- Security-> SSL certificate and key management -> Manage endpoint security configurations -> *endpoint_configuration_name* -> SSL configurations -> *SSL_configuration_name* -> Quality of protection settings (QoP)

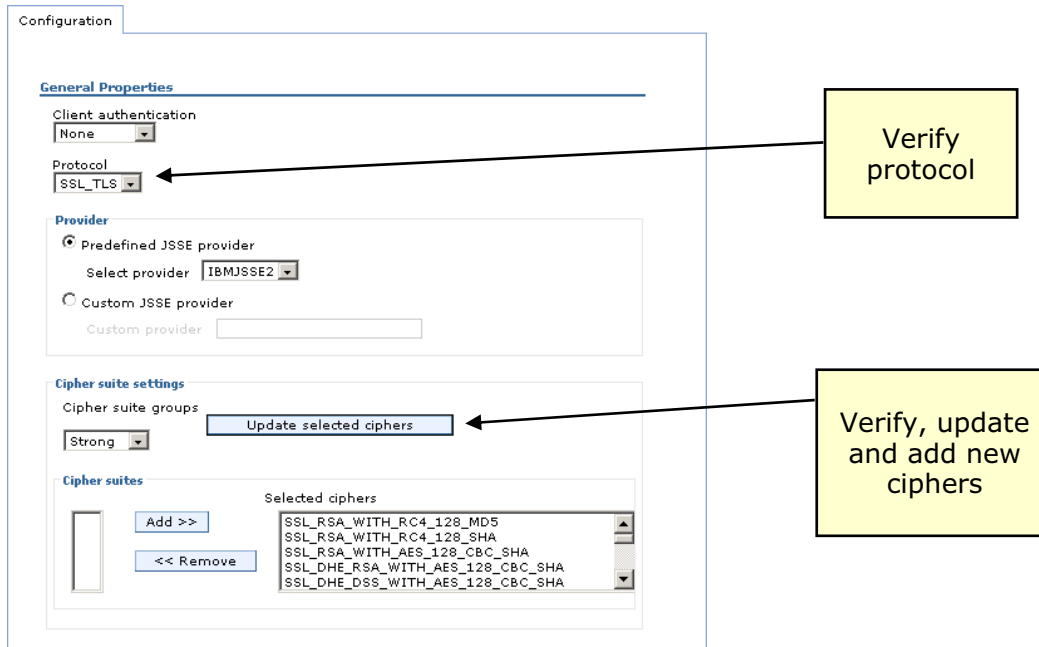


Figure 12-29. SSL problems – Handshake failures (3 of 3)

WA5711.0

Notes:

Client authentication - Specifies the whether SSL client authentication should be requested if the SSL connection is used for the server side of the connection.

If **None** is selected, the server does not request that a client certificate be sent during the handshake. If **Supported** is selected, the server requests that a client certificate be sent. If the client does not have a certificate, the handshake might still succeed. If **Required** is selected, the server requests that a client certificate be sent. If the client does not have a certificate, the handshake fails.

Protocol - Specifies the Secure Sockets Layer (SSL) handshake protocol. This protocol is typically SSL_TLS, which supports all handshake protocols except for SSLv2 on the server side. When United States Federal Information Processing standard (FIPS) option is enabled, Transport Layer Security (TLS) is automatically used regardless of this setting.

Selected ciphers - Specifies the ciphers that are effective when the configuration is saved. These ciphers are used to negotiate with the remote side of the connection during the handshake. A common cipher needs to be selected or the handshake fails.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SSL error messages

- The Java Cryptographic Extension (JCE) files were not found
 - Error when launching IKeyman
- Unable to verify MAC (message authentication code)
 - Error when the wrong keystore password is used

```
CWPKI0033E: The keystore located at
"C:/WebSphere/AppServer/profiles/profile1/etc/trust.p12"
failed to load due to the following error: Unable to
verify MAC.
```

Figure 12-30. SSL error messages

WA5711.0

Notes:

The Java Cryptographic Extension (JCE) files were not found

To resolve this problem:

1. Set the `JAVA_HOME` parameter so that it points to the Java Developer Kit that is shipped with WebSphere Application Server.
2. Rename the file `install_dir/java/jre/lib/ext/gskikm.jar` to `gskikm.jar.org`.

Unable to verify MAC

To resolve this problem:

1. Change the **Password** field that references this keystore by using the correct password. The default password is `WebAS`

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Common SSL connection error message

- SSL handshake failure
 - Error when no trusted certificate is found
 - The certificate alias cannot be found in the keystore

```
CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN
"CN=192.168.1.204, O=IBM, C=US" was sent from target host:port
"192.168.1.12:9403".
```

```
The signer may need to be added to local trust store
"c:\WASV61\profiles\Dmgr01\etc\trust.p12" located in SSL configuration
alias "DefaultSSLSettings" loaded from SSL configuration
file:c:\WAS\App\profiles\Dmgr01\properties\ssl.client.props".
```

```
The extended error message from the SSL handshake exception is: "No
trusted certificate found".
```

Figure 12-31. Common SSL connection error message

WA5711.0

Notes:

The message above shows an example of one of the most common errors that occurs when attempting to establish an SSL connection from a client to a server. In this case, the server has sent a certificate that is not recognized by the client; that is, the trust store of the client does not contain the corresponding signer. The error message provides detailed information on this error, which should make it easier to correct. Notice that it indicates the host:port, the missing signer, the SSL configuration, and even the trust store that is being used. This tells the administrator precisely what needs to be done: In this case, the missing signer needs to be obtained and added to the **trust.p12** trust store specified.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SSL problem determination steps

- Check the SystemOut.log and SystemErr.log files to determine which SSL problem you are facing

- Use the search facility within ISA or through the WebSphere Support site to look for common solutions

- If a common solution is not found, gather a JSSE trace
 - Set the following system property and restart the application server
 - -Djavax.net.debug=true
 - Recreate the problem and view trace output in the SystemOut.log
 - Look for exceptions/stack traces and work back up the thread to find the lines preceding the exception.
 - The inputs like client key, server key, expiration dates, and ciphers are listed in the trace output.

Figure 12-32. SSL problem determination steps

WA5711.0

Notes:

Set the system property on the client and server processes: `-Djavax.net.debug=true`. For the server, add the system property to the Generic JVM arguments property of the Java virtual machine settings page.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Java 2 security problems (1 of 2)

- Java 2 security access control exceptions at run time may result if:
 - An application is not prepared for Java 2 security
 - The application provider does not provide a *was.policy* file as part of the application
 - The application provider does not communicate the expected permissions

- Gather diagnostic data from the SystemOut.log file
 - Set the `com.ibm.websphere.java2secman.norethrow` property for the server
 - The AccessControl exception contains the
 - Permission violation that causes the exception
 - Exception call stack
 - Permissions granted to each stack frame

Notes:

If you suspect there is a problem with Java 2 security, try to answer the following questions.

1. Has the application been designed with Java 2 security in mind?
2. What operating system APIs or system files does your application need to access?
3. What permissions have you granted your application? (*was.policy*)
4. Did you manually edit the property file or use the *install_root/java/jre/bin/policytool*?

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Java 2 security problems (2 of 2)

- Handling Java 2 security access exceptions
 - Disable Java 2 security—easy, but will organization security policies allow it?
 - Leave Java 2 security enabled, but then grant
 - Either just enough additional permissions
or
 - All permissions to just the problematic application
- Use the PolicyTool (`<WAS_ROOT>/java/jre/bin/policytool`) to edit policy files
 - Grant the **java.security.AllPermission** permission in the **was.policy** file that is embedded in the application EAR file
- For example:

```
grant codeBase "file:${application}" { permission  
java.security.AllPermission; };
```

Figure 12-34. Java 2 security problems (2 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative console runtime messages, log file messages, and codes

After completing this topic, you should be able to:

- Identify the different types of WebSphere and security-related messages and codes

Figure 12-35. Administrative console runtime messages, log file messages, and codes

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Security Association Service messages

- JSAS messages are from Security Association Service
 - Examples

```
JSAS0201E: [{0}] Invocation credential realm does not match target's realm: {0}. If using the SWAM authentication mechanism, you should switch to using LTPA instead for remote IIOP invocations.  
Explanation: Attempting a remote invocation over IIOP using the SWAM authentication mechanism is not supported.
```

```
User Response: Retry with the LTPA authentication mechanism configured in Global Security
```

```
JSAS0202E: [{0}] Credential token expired. {1}  
Explanation: The credential token associated with the user credential has expired. This typically occurs with LTPA.
```

```
User Response: Close the client and login again.
```

Figure 12-36. Security Association Service messages

WA5711.0

Notes:

When these messages occur they can be viewed in the administrative console.

Select **Troubleshooting -> Runtime Messages** and then your choice of **Error**, **Warning**, or **Information**.

Each system message has a unique message identifier that is nine characters in length and is in the form `cccc1234x`. The first four characters (`cccc`) indicate the WebSphere Application Server component that issued the message. The next four characters (`1234`) indicate the specific message that is being issued by the component. The last character (`x`) indicates the severity of the message. Its value is either `I` (informational), `w` (warning), or `E` (error).

Example: `JSAS0201E?JSAS=component, 0201E=specific message, E=error level severity`

A complete list of JSAS messages can be found in the WebSphere V6.1 information center.

Select **Network Deployment, Version 6.1 -> Reference -> Messages -> JSAS**

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere Security messages

- SECJ messages are from WebSphere Security
 - Examples

```
SECJ0007E: Error during security initialization.  
The exception is {0}.
```

```
Explanation: An unexpected error occurred during  
security initialization.
```

```
User Response: This is a general error. Look for  
previous messages that may be related to the  
failure or a configuration problem. Enabling  
security debug trace for components  
com.ibm.ws.security.* and com.ibm.ejs.security.*  
may yield additional information.
```

```
SECJ0056E: Authentication failed for reason {0}
```

```
Explanation: Authentication failed with the  
specified reason.
```

```
User Response: Verify that the user id and password  
are entered correctly. Consult with the  
administrator of the user registry if the problem  
persist.
```

Figure 12-37. WebSphere Security messages

WA5711.0

Notes:

A complete list of SECJ messages can be found in the WebSphere V6.1 information center.

Select **Network Deployment, Version 6.1 -> Reference -> Messages -> SECJ**

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web UI Security Center messages

- SECG messages are from Web UI Security Center
 - Examples

SECG0005E: An exception occurred when exporting Lightweight Third Party Authentication (LTPA) keys: The exception is {0}.

Explanation: Unable to get the Lightweight Third Party Authentication (LTPA) keys from the server.

User Response: Regenerate the keys and try the operation again

SECG0027E: The Ignore case option is required for the Lightweight Directory Access Protocol (LDAP) directory type {0}.

Explanation: Select the Ignore case option for the LDAP directory type selected.

User Response: Enable the Ignore case option in the administrative console. Expand Security > User Registries. Click LDAP and select the Ignore case option.

Figure 12-38. Web UI Security Center messages

WA5711.0

Notes:

A complete list of SECG messages can be found in the WebSphere V6.1 information center.

Select **Network Deployment, Version 6.1 -> Reference -> Messages -> SECG**

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web Services Security (WS-Security) messages

- WSEC messages are from Web Services Security
 - Examples

```
WSEC0001E: Error trying to find Security Server.
The exception is {0}.
Explanation: This exception is unexpected. The
cause is not immediately known.
User Response: If the problem persists, see problem
determination information on the WebSphere
Application Server Support...

WSEC0007W: Server level Web Services Security
configuration file {0} is not found.
Explanation: The server level Web Services Security
configuration document might be corrupted or
missing. The file provides the default binding
configuration for Web Services Security.
User Response: If you would like to use the default
bindings information, please copy ws-security.xml
from the ${USER_INSTALL_ROOT}/config/templates
directory.
```

Figure 12-39. Web Services Security (WS-Security) messages

WA5711.0

Notes:

A complete list of WSEC messages can be found in the WebSphere V6.1 information center.

Select **Network Deployment, Version 6.1 -> Reference -> Messages -> WSEC**

There are two other important sets of security-related messages, JSSL and WSSK.

JSSL is ORB SSL Extensions messages.

WSSK is Web Services Security Kerberos (WS-Security Kerberos) messages. A complete list of these messages can be found in the WebSphere V6.1 information center.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Virtual member manager (VMM) messages

- CWWIM messages are from the virtual member manager
 - Examples

```
CWWIM0000E: The virtual member manager
configuration XML file 'file_name' was not found.
Explanation: Virtual member manager cannot run
without the configuration information. Virtual
member manager cannot continue without this file.
The name of this file is wimconfig.xml.The path of
this file is VMM_HOME/config...
Programmer Response: Ensure that this file exists.
```

```
CWWIM0500E: The 'realm_name' realm name specified
is not valid.
Explanation: The realm name is not defined in the
virtual member manager configuration file.
Programmer Response: Ensure that the realm name
exists and that it is spelled correctly. If it does
not exist, you must define the realm or specify a
different, valid realm.
```

Figure 12-40. Virtual member manager (VMM) messages

WA5711.0

Notes:

A complete list of CWWIM messages can be found in the WebSphere V6.1 information center.

Select **Network Deployment, Version 6.1 -> Reference -> Messages -> Virtual member manager messages**

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Additional tools and techniques

After completing this topic, you should be able to:

- Locate and validate security configuration files
- Track LTPA tokens
- Disable global security

Figure 12-41. Additional tools and techniques

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative console security PD tools

- **Test connection** button attempts to connect to LDAP server from deployment manager using LDAP server host name and port
- **SSL certificate and key management**
 - Certificate expiration
 - End point SSL configuration
 - Keystores and certificates
 - Based on IKeyMan functionality from previous WebSphere versions
 - Key managers
 - Trust managers

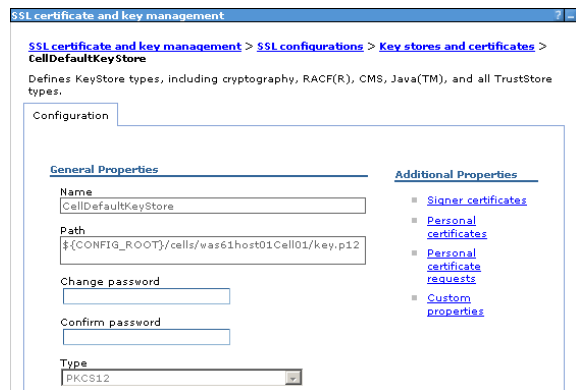
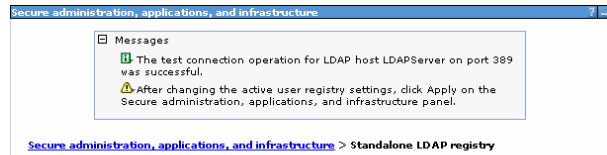


Figure 12-42. Administrative console security PD tools

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Security configuration files (1 of 2)

- **security.xml**
 - Each profile has a copy of security.xml of the deployment manager located at `<WAS_HOME>\profiles\<Profile_Name>\config\cells\<Cell_Name>`
 - Contains all of the security configuration information and status
 - User registry
 - Authentication mechanism
 - Many more
 - Each server has its own copy of security.xml which may override the cell-wide configuration
 - Enable/disable application security
 - Enable/disable Java 2 security
- **ws-security.xml**
 - Each application server has a copy of the ws-security.xml file, which defines the default binding information for Web services security
- **sas.client.props**
 - Each profile has a copy of sas.clients.props in its properties directory
 - Contains client-side properties used by Secure Association Services
- **App.policy and was.policy**
 - Contain policy information for Java 2 Security
 - Each profile has a copy of app.policy in its properties directory
 - Each application has a copy of was.policy in its EAR file

Figure 12-43. Security configuration files (1 of 2)

WA5711.0

Notes:

A server copy of security.xml is located at

```
<WAS_HOME>\profiles\<profile_name>\config\cells\<cell_name>\nodes\
<node_name>\servers\<server_name>
```

- **sas.server.props** -SAS Server properties file (now disabled, use administrative console or security.xml instead). Properties which used to be specified in this file are now configured using security.xml.
- **soap.client.props** -In order to avoid having to expose user ID and password information on the command line when invoking the wsadmin tool, set the appropriate values for `com.ibm.SOAP.loginUserId=<userid>` and `com.ibm.SOAP.loginPassword=<passwd>`

Instructor notes:

Purpose — Identify the major security configuration files and their location. Point out that these files should almost never be edited manually. The administrative clients should be used to write the configuration to these files. However, in some situations it may not be possible to start a server, requiring the files to be manually edited.

Details — **sas.client.props** is used for RMI/IIOP (typically EJB calls), not for HTTP/Web calls.

sas.client.props might be used by one application server when it calls another (acting as a client of the second application server).

sas.client.props is also used by external EJB client program (not application servers), like wsadmin, or application clients. In that case, a common problem is when the client program cannot find the **sas.client.props** file, or finds the wrong one.

Additional information —

Transition statement —

Security configuration files (2 of 2)

- wimconfig.xml
 - Each profile has a copy of wimconfig.xml located at
`<WAS_HOME>\profiles\<Profile_Name>\config\cells\<Cell_Name>
/wim/config`
 - Contains all of the virtual member manager configuration information

- `<WAS_HOME>\etc\wim`
 - This directory contains the virtual member manager setup and migration files

- fileRegistry.xml
 - If file-based user registry is configured, each profile has a copy of fileRegistry.xml located at
`<WAS_HOME>\profiles\<Profile_Name>\config\cells\<Cell_Name>`
 - This is the file repository and it contains:
 - User and group identifiers
 - Encrypted passwords for users

Figure 12-44. Security configuration files (2 of 2)

WA5711.0

Notes:

In the fileRegistry.xml file the passwords are encrypted using a one-way hash by applying the message digest algorithm specified in the VMM configuration file (wimconfig.xml). The default value for the message digest algorithm is SHA-1. The algorithm can be changed using the wsadmin command `$AdminTask updateIdMgrFileRepository`.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Tools to validate the security configuration

- Various tools can be downloaded from the IBM Support site for checking and validating security configuration
- These tools include:
 - ACert
 - A command line tool that checks expiration dates of all SSL certificates defined in WebSphere Application Server SSL repositories
 - The expiration dates of each certificate are displayed
 - WebSphere Security Scanning Tool (WSST)
 - A command line tool that scans static WebSphere Application Server security configuration files to look for potential vulnerabilities
 - The tool produces an HTML report that contains
 - Security configuration checks performed
 - Status of each check
 - Corrective action if necessary
 - A link to the information center task related to the corrective action
- LDAP browsers and editors
 - Very useful for working with LDAP user registries

Figure 12-45. Tools to validate the security configuration

WA5711.0

Notes:

These tools can be found by going to the Web site www.ibm.com/support/us. Select the link Downloads and drivers. Search on Category: WebSphere, Sub-category: WebSphere Application Server. Then search on the tool by name.

ACert - Checking SSL certificates can help avoid application failures due to expired SSL certificates used for authentication within WebSphere Application Server and secure communications between the Application Server and the plug-in running within a Web server.

ACert does not check the WebSphere truststore for expired certificates, only SSL certificates that are defined in SSL repertoires.



Important

WSST does not check for runtime penetration vulnerabilities.

The tool is not a general purpose WebSphere Application Server configuration diagnostic tool intended to aid in the problem determination of configuration problems.

The tool is not a fail safe guarantee that system is totally secure.

The tool does not do network, host, physical, or operating system security vulnerability analysis.

LDAP Browsers - Numerous free LDAP browsers are available for download. A very useful free Java browser/editor can be downloaded from the Argonne National Laboratory site, <http://www-unix.mcs.anl.gov/~gawor/ldap/download.html>

Instructor notes:

Purpose — Point out the existence of these two well-known tools. Also advise the students to use the IBM Support Assistant to search for other tools that are available, or that may become available, that can be helpful for security troubleshooting.

Details —

Additional information —

Transition statement —

Tracking LTPA tokens

- To track LTPA tokens, configure your Web browser to warn about cookies
- If you do not get a warning after a successful authentication you may have problems with
 - Domain suffix for the LTPA SSO configuration
 - Proxy server configuration

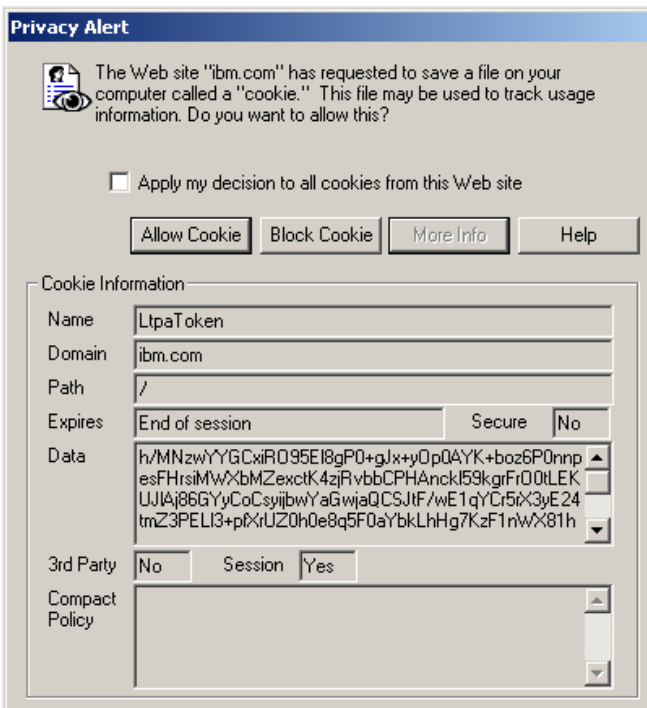


Figure 12-46. Tracking LTPA tokens

WA5711.0

Notes:

You can enable Microsoft Internet Explorer to warn about cookies as follows: Select **Tools -> Internet Options -> Privacy tab -> Advanced**. Select **Override automatic cookie handling**, and select **Prompt**.

It is useful to determine how WebSphere Application Server is managing the LTPA cookie or token. You should enable your Web browser to warn about cookies. Once you do this, your browser will inform you when WebSphere Application Server sends back an LTPA token. If you do not get such a warning after a seemingly successful authentication, the browser or some intermediate proxy has probably swallowed the cookie. It could be that the DNS domain setting in the LTPA SSO page is wrong or that a proxy server is configured improperly. It is often helpful to turn on the Web server plug-in tracing and possibly the WebSphere Application Server Web container tracing (com.ibm.ws.Webcontainer.*) to see if the LTPA token is being generated by WebSphere Application Server and then perhaps lost.

Known issues on interactions between security proxies and LTPA

1. If you login to a proxy such as TAM, it will assert an identity to WebSphere, which will create an LTPA cookie. If you later log into TAM from that same browser as a different user, and the LTPA cookie has not expired, you will be one user to TAM, and a different user to WebSphere. The fact that the LTPA exists and is valid prevents the TAI from being invoked and asserting the new identity.
2. Another LTPA issue can arise when the same browser goes to two different domains and gets two different LTPA cookies. If an application in one domain explicitly logs out using a post to `ibm_security_logout`, only the LTPA cookie for that domain will be removed, so if the new user then navigates to the application in the other domain, they will be logged in as the other user. This again is most likely an issue where there is a security proxy hiding that fact that multiple domains exist.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Disabling administrative security

- Sometimes it is necessary to disable administrative security in order to troubleshoot security-related problems
- If the application server or deployment manager is running, use the administrative Console
 1. Select **Security -> Secure administration**
 2. Clear **Enable administrative security**
 3. Save changes to the master repository
- If the application server cannot be started, for example
 - Password of the server user ID in the user registry is expired or
 - The user registry cannot be reached for authentication
 - Disable administrative security using the command line
 1. `<WAS_HOME>\bin\wsadmin.bat -conntype NONE`
 2. `wsadmin>securityoff`
 3. `wsadmin>quit`
 4. Start `server1` or `dmgr`
 - **Note:** Only changes the `security.xml` file of the local node

Figure 12-47. Disabling administrative security

WA5711.0

Notes:

The above procedure should work without any problem. However, in case it fails, you could try to disable Global Security by directly editing the file

`<WebSphere_home>\profiles\<profilePath>\config\cells\<cell_name>\security.xml`, and changing the security attribute `enabled="true"` to `enabled="false"`.

Some other properties, like enforcing Java 2 security, can also be found in this file. Care should be taken when modifying this file directly.

Instructor notes:

Purpose — Emphasize that the slide shows the recommended ways to disable security, but the method of manually editing is also provided in the notes.

Details —

Additional information —

Transition statement —

Security references

- *IBM WebSphere Application Server V6.1 Security Handbook*, SG24-6316-01
- “WebSphere Application Server V6 advanced security hardening - Part 1”
 - http://www.ibm.com/developerworks/websphere/techjournal//0512_botzum/0512_botzum1.html
- “WebSphere Application Server V6 advanced security hardening - Part 2”
 - http://www.ibm.com/developerworks/websphere/techjournal//0512_botzum/0512_botzum2.html
- *IBM WebSphere Developer Technical Journal: Using the Java Secure Socket Extension in WebSphere Application Server*
 - http://www.ibm.com/developerworks/websphere/techjournal/0502_benantar/0502_benantar.html

Figure 12-48. Security references

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. Which two application server components have a security collaborator process?
2. In which log files would you most likely find stack traces resulting from security-related exceptions?
3. Describe how you can get more detailed information about WebSphere security components written to a log file?
4. Which configuration file contains the global security information including: security status, user registry, and authentication mechanisms?

Figure 12-49. Checkpoint

WA5711.0

Notes:

Write down your answers here:

- 1.
- 2.
- 3.
- 4.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

1. Which two application server components have a security collaborator process?
 - **The Web container and the EJB container each have a security collaborator.**
2. In which log files would you most likely find stack traces resulting from security-related exceptions?
 - **The SystemOut.log and SystemErr.log would contain stack traces resulting from security-related exceptions.**
3. Describe how you can get more detailed information about WebSphere security components written to a log file?
 - **For each application server, node agent, and deployment manager you can enable tracing of security components at different levels of detail. This information can be written to a trace.log file.**
4. Which configuration file contains the global security information including: security status, user registry, and authentication mechanisms?
 - **The security.xml file contains all of the global security configuration information.**

Figure 12-50. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe common problems with WebSphere security
- Recognize symptoms of common security-related problems
- Analyze relevant log files for security messages
- Enable server tracing on relevant security components
- Analyze and interpret trace information
- Locate the security configuration files
- Use tools to validate the security configuration files

Figure 12-51. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 7: Troubleshooting security problems

Figure 12-52. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 13. Application deployment problems

Estimated time

00:30

What this unit is about

This unit looks at common problems that can arise while deploying an application to WebSphere Application Server V6.1. Such problems include:

What you should be able to do

After completing this unit, you should be able to:

- Detect application deployment problems
- Examine the WebSphere Application Server logs
- Enable tracing of the appropriate components and interpret the data
- Use wsadmin to test and reproduce a problem
- Use AST to validate application configuration files
- Describe the Class Loader Viewer tool

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

- WebSphere Application Server Network Deployment, version 6.1 information center: “Developing and deploying applications”
- ftp://ftp.software.ibm.com/software/webserver/appserv/library/v61/wasv610nd_devdep.pdf Chapter 9 “Troubleshooting deployment”

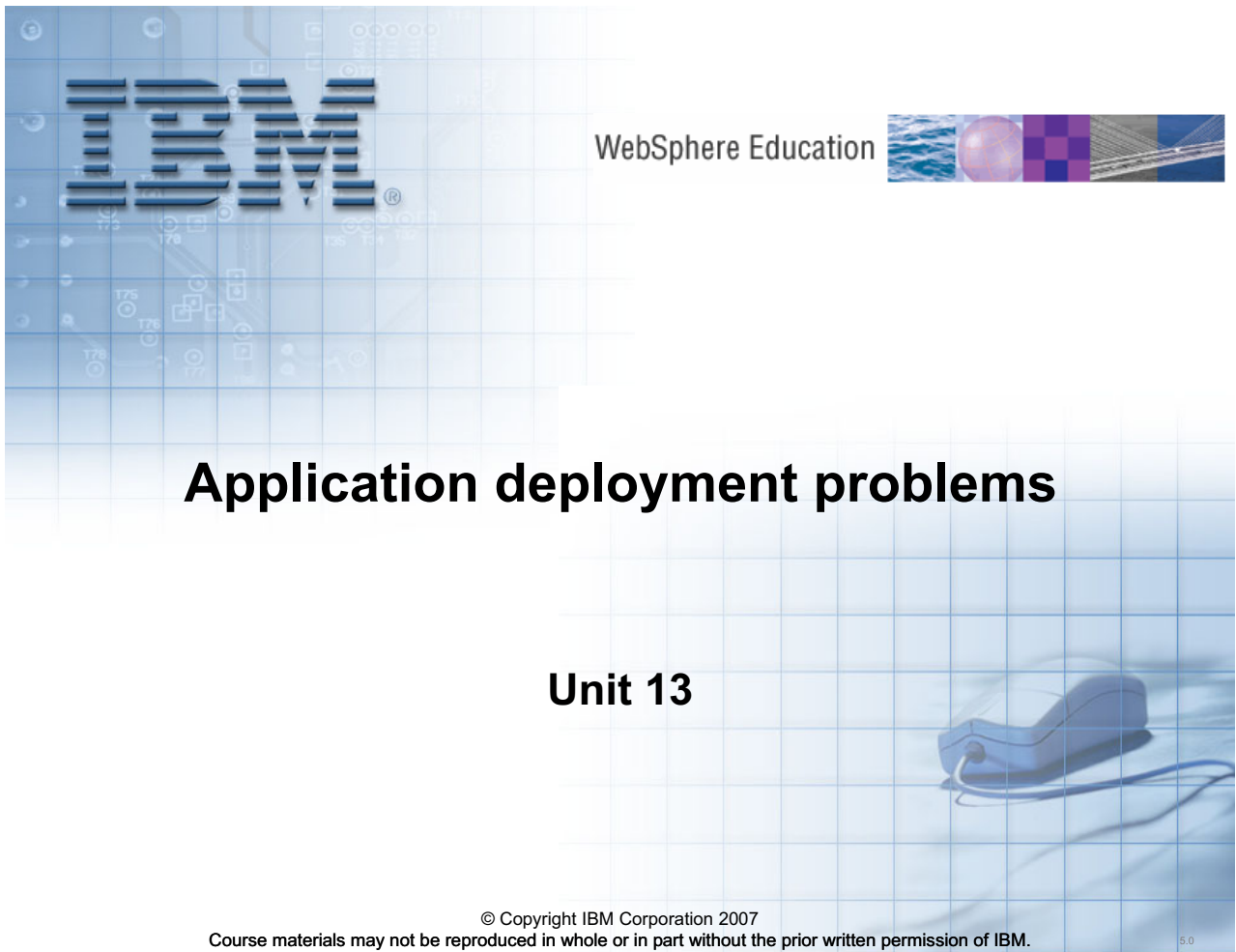


Figure 13-1. Application deployment problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Detect an application deployment issue in WebSphere Application Server V6.1
- Determine the root cause of an application deployment problem by using the WebSphere Application Server V6.1 log files
- Resolve the application deployment issue by using the administrative console and the Application Server Toolkit (AST)
- Enable on WebSphere servers and `wsadmin` during deployment process in order to debug a deployment problem
- Identify and resolve class loader issues using Class Loader Viewer

© Copyright IBM Corporation 2007

Figure 13-2. Unit objectives

WA5711.0

Notes:

The value of WebSphere Application Server V6.1 is in the applications that it hosts. Therefore, the successful deployment of an application is of paramount importance. WebSphere Application Server V6.1 provides a great amount of flexibility in the application deployment process by providing the `wsadmin` script interpreter command line interface and the administrative console Web browser-based application installation wizard. There are other application installation methods, such as Java rapid deployment and application deployment using a JSR-88 compliant deployment manager. However, this unit focuses on the `wsadmin` script interpreter and the administrative console wizard, only.

Using the application installation wizard or the `wsadmin` script interpreter is usually performed by a WebSphere Application Server V6.1 administrator, a WebSphere Application Server V6.1 operator, an application developer, or a systems integration specialist (in most organizations a single person will have more than one of these roles). Typically, an application will be developed and tested in a WebSphere Application Server V6.1 test environment prior to deployment into production. In most test environments an effort is made to have the middleware mirror the production target as closely as possible. Usually, the same level of hardware and software can be used in the test environment, but

other aspects of the production environment, such as routers, firewalls, proxies, and so on, will not be present in the test system. As a result, unforeseen connectivity and configuration issues can occur when the application is rolled out to the production environment.

Instructor notes:

Purpose —

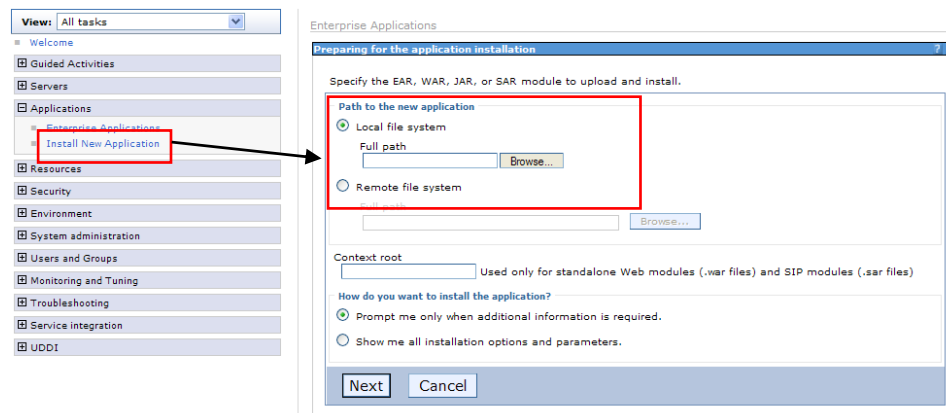
Details — Mention to the students that there is a exercise associated with this module.

Additional information —

Transition statement —

Application installation methods

- The administrative console application installation wizard guides the user through the installation process



- *wsadmin* can run interactively or be used to run installation scripts from the command line
 - `wsadmin -f <jacl or jython file>`
 - `wsadmin $AdminApp install <application.ear>`
 - `wsadmin $AdminApp installInteractive`

© Copyright IBM Corporation 2007

Figure 13-3. Application installation methods

WA5711.0

Notes:

An application can be installed into WebSphere using two primary methods:

1. WebSphere administrative console installation wizard by navigating through
 - Applications -> Enterprise Applications -> Install**
 - Applications -> Install New Application**

- *wsadmin* script interpreter

```
wsadmin $AdminApp install
```

The administrative console and *wsadmin* can also be used to update running applications

Uninstall old EAR file and install updated EAR file

An application deployment issue can manifest itself as:

1. A warning during the deployment process

2. An error while the application information is being persisted to the WebSphere Application Server V6.1 configuration files.
3. A failed request when the application is tested.

In any of these cases the result is the same: The application will not work.

During the installation of the application, exceptions can occur as a result of a failure by the installation process. Once the application is installed, failure can occur while trying to start the application, or once the application is started and an actual transaction or request is sent to the application.

The IBM Redbook *WebSphere Application Server Network Deployment, version 6: Developing and deploying applications* is an excellent reference for the material discussed in this unit. It is available from

ftp://ftp.software.ibm.com/software/webserver/appserv/library/v61/wasv610nd_devdep.pdf
as part of the WebSphere Application Server V6.1 information center.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Problem areas in application installation

- During the application deployment process, problems might be encountered during any of the following activities:
 - Installing, deploying, or updating applications
 - Warning or error messages about application components
 - Corrupt EAR file
 - Starting the application for the first time
 - Application fails to start with error message
 - Namespace problems
 - Running the application for the first time
 - Application client reports error condition
 - Application does not work as expected
 - Data source problems
 - Resource connectivity issues
 - Namespace and scoping issues

© Copyright IBM Corporation 2007

Figure 13-4. Problem areas in application installation

WA5711.0

Notes:

During application deployment configuration values such as JNDI names and security information might need to be provided. If any of the values specified are incorrect, they might not be detected until the application is started or run for the first time. This is also true if incorrect credentials or method permissions are specified. They will not be detected until the resource they protect is accessed.

If any problems are detected during the deployment process, then one of the following will occur:

- An error will prevent the application from being deployed.
- A warning will not prevent the application from being deployed but may require correction before the application will run correctly.

However, the absence of an error or warning does not guarantee that the application will function correctly. As stated above, some errors will not be detected until the application is started or run for the first time.

Instructor notes:

Purpose —

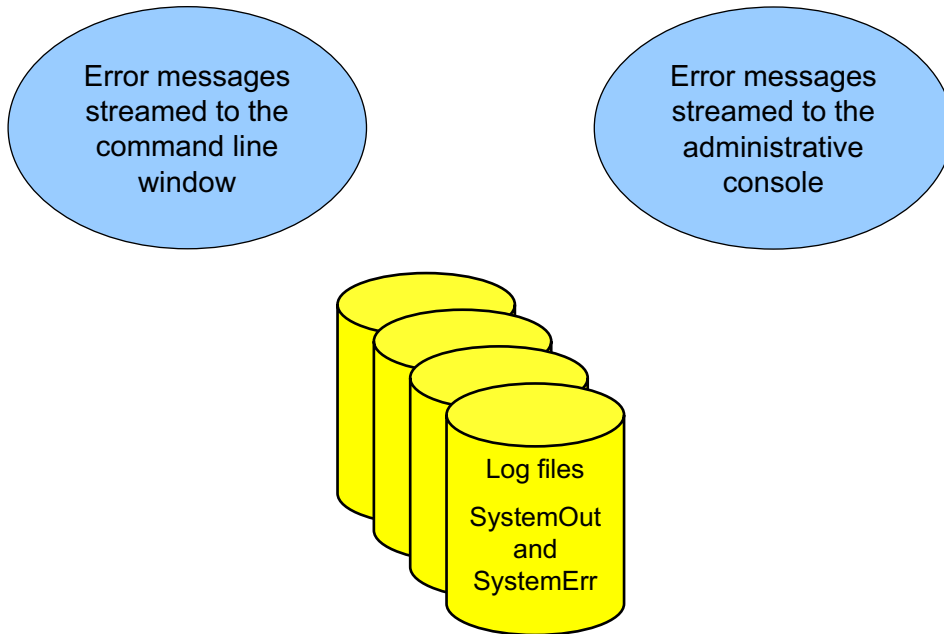
Details —

Additional information —

Transition statement —

Detecting an application installation issue

- Collect diagnostic data



© Copyright IBM Corporation 2007

Figure 13-5. Detecting an application installation issue

WA5711.0

Notes:

Detecting application installation failures can be done through the error messages written to the administrative console, when using the administrative console installation wizard, or the command line window, when using `wsadmin`. If an application fails to start its state will be listed as `STOPPED` or `UNAVAILABLE`. This can be verified by running the `serverStatus`. [bat | sh] script for that particular server.

From a user's perspective, an application failure could show up on the Web browser as an HTTP failure. However, it is more likely to show up as an error page that has been designed to inform users that the current site is unavailable.

Some examples of installation failure and their results are:

- Installing an application but it does not show up in the Enterprise Applications list of the administrative console
- The application fails when you attempt to Save to Master Configuration after the installation is complete
- EJBs do not deploy or a JSP compiler error occurs

- A *NameNotFoundException* occurs during installation
- Failure to load resource WEB-INF/ibm-web-bnd.xmi in archive file

In some cases, the application will actually be started but it will not work from end-to-end, through the Web server. This is generally a problem with the plug-in configuration. Failing to map the modules to both Web server and application server during the application deploy process will prohibit the addition of new application related URI entries in to the generated plug-in-cfg.xml on the targeted Web server, which in turn will not allow any traffic to the new enterprise application using Web server. This additional mapping of application modules to the Web server definition is only required from V6.0 onwards. Below is one of the examples which explains a classical customer situation who has only mapped to application server, but not to the Web server.

For example, the application is accessible through the application server transport:

- <http://www.appserverhostname.com:9080/homepage>
- <https://www.appserverhostname.com:9044/homepage>

But not through the Web server:

- <http://www.webserverhostname/homepage>
- <https://www.webserverhostname/homepage>

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative console Troubleshooting menu

Runtime Events
Use this page to view runtime events that propagate from the server.

Runtime Events have been disabled by default ("None"). To enable a event level please select from the list. "Error" would enable only Error runtime events. "Warning" would enable both Error and Warning runtime events. Info would enable all runtime events.

Error

Apply

Preferences

Timestamp	Message Originator	Message
Oct 9, 2007 1:18:51 PM EDT	com.ibm.ws.console.core.mbean.MBeanHelper	Could not invoke an operation on object: WebSphere
Oct 9, 2007 1:18:49 PM EDT	com.ibm.ws.runtime.component.EJBContainerImpl	WSVR0040E: addEjbModule failed for plantsby/WebSphe
Oct 9, 2007 1:18:49 PM EDT	com.ibm.ws.runtime.component.EJBContainerImpl	WSVR0046E: Unable to bind, plantsby/ShoppingCartHo
Oct 9, 2007 1:18:49 PM EDT	com.ibm.ws.runtime.component.EJBContainerImpl	WSVR0046E: Unable to bind, plantsby/CatalogHome: p
Oct 9, 2007 1:18:49 PM EDT	com.ibm.ws.runtime.component.EJBContainerImpl	WSVR0046E: Unable to bind, plantsby/LoginHome: pla
Total 5		

The **Configuration Validation** and **Runtime Messages** functions are useful for detecting if an error has occurred

© Copyright IBM Corporation 2007

Figure 13-6. Administrative console Troubleshooting menu

WA5711.0

Notes:

The *Configuration Validation* and *Runtime Messages* functions are excellent places to detect if an error has occurred. The messages are time sorted and grouped by severity and provide immediate feedback regarding the state of the system after performing an application installation. Both the *Configuration Validation* messages and the *Runtime Messages* are grouped by *Error*, *Warning*, and *Information*.

Runtime messages are formatted across columns containing the timestamp, message originator and the actual message. The message is a hyperlink leading to additional detailed information.

Instructor notes:

Purpose —

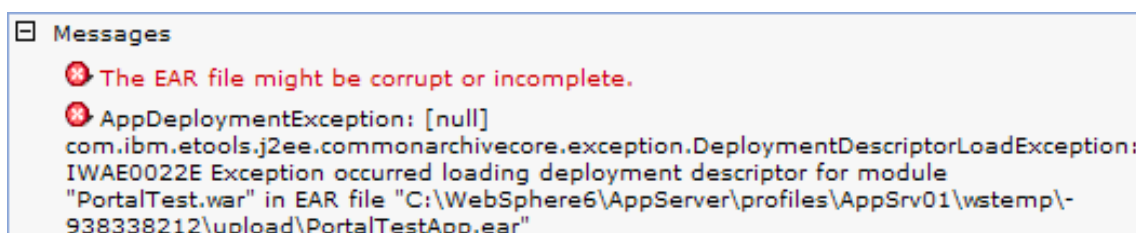
Details —

Additional information —

Transition statement —

Examining console and error log file messages

- During the installation process, any exception that occurs will be logged to the **SystemErr.log** file
- If you are using the application installation wizard of the administrative console, the errors will be displayed to the administrative console
 - For example, the following error message is a result of a corrupted or incorrect application WAR file:



© Copyright IBM Corporation 2007

Figure 13-7. Examining console and error log file messages

WA5711.0

Notes:

Application deployment problems, like all WebSphere Application Server V6.1 errors, warnings and information messages will be logged to the system logs based on the level of trace specified for the system. By default, any error that occurs will be captured in the SystemErr.log or the SystemOut.log (or both). If you are using the administrative console, an error will be rendered to the panel that is currently active as soon as the error, warning or informational message is generated. The *Troubleshooting* section of the administrative console navigation menu contains links that guide the user to the errors, warnings and informational messages that can occur as a result of configuration or runtime activities. If you are using the *wsadmin* script interpreter to install an application, then error, warning and informational messages will be streamed to the command console as soon as they are generated. Receiving an error or warning message is an indication that the log files should be inspected for further information about the event that just occurred.

Some application deployment problems are simple syntax problems. For example, invalid characters, such as - / \ : * --> --> < > | in the name of the application can lead to an error during installation. Check to verify that you are not using any special characters for your application name.

Another common problem is user error. The application installation might complete correctly, using the *wsadmin* script interpreter, but when you look for your application in the administrative console, it does not show up. This is a result of not having run the *save* command after the installation process. As more experience is gained with the WebSphere Application Server V6.1 installation process, user error occurs less frequently.

Instructor notes:

Purpose —

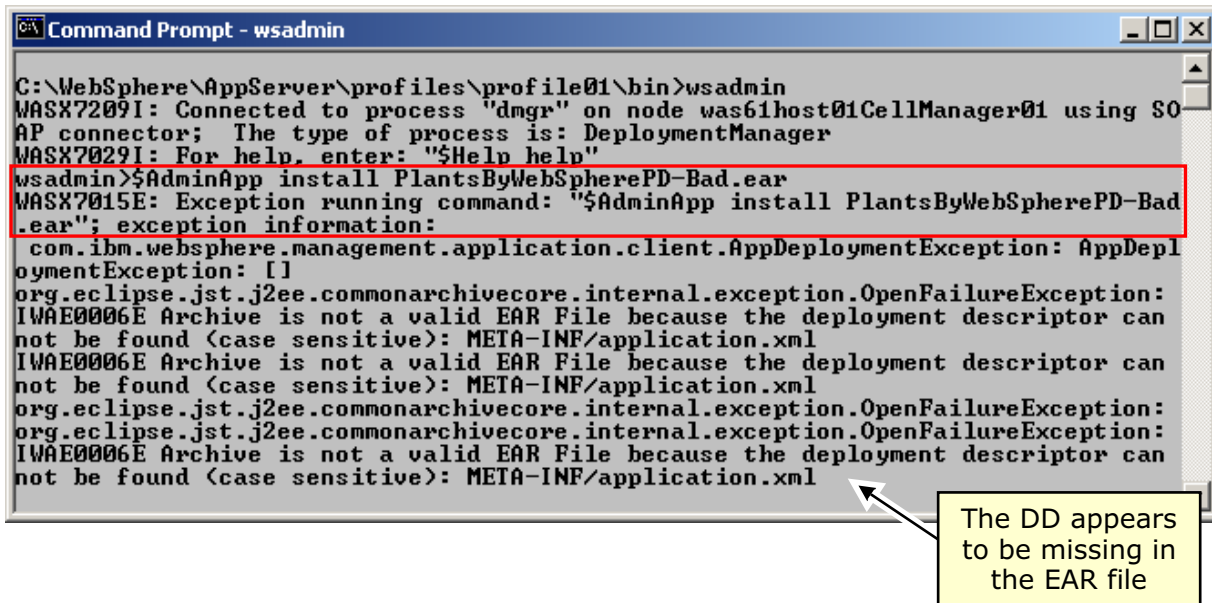
Details —

Additional information —

Transition statement —

wsadmin command line exceptions

- Installing an EAR file from the command line using wsadmin might result in the following exceptions



```
Command Prompt - wsadmin
C:\WebSphere\AppServer\profiles\profile01\bin>wsadmin
WASX7209I: Connected to process "dmgr" on node was61host01CellManager01 using S0
AP connector; The type of process is: DeploymentManager
WASX7029I: For help, enter: "$Help help"
wsadmin>$AdminApp install PlantsByWebSpherePD-Bad.ear
WASX7015E: Exception running command: "$AdminApp install PlantsByWebSpherePD-Bad
.ear"; exception information:
  com.ibm.websphere.management.application.client.AppDeploymentException: AppDepl
oymentException: [ ]
  org.eclipse.jst.j2ee.commonarchivecore.internal.exception.OpenFailureException:
  IWAE0006E Archive is not a valid EAR File because the deployment descriptor can
  not be found (case sensitive): META-INF/application.xml
  IWAE0006E Archive is not a valid EAR File because the deployment descriptor can
  not be found (case sensitive): META-INF/application.xml
  org.eclipse.jst.j2ee.commonarchivecore.internal.exception.OpenFailureException:
  org.eclipse.jst.j2ee.commonarchivecore.internal.exception.OpenFailureException:
  IWAE0006E Archive is not a valid EAR File because the deployment descriptor can
  not be found (case sensitive): META-INF/application.xml
```

The DD appears to be missing in the EAR file

© Copyright IBM Corporation 2007

Figure 13-8. wsadmin command line exceptions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Typical application installation errors

- While installing, deploying, or updating the application, the following errors might be encountered:
 - Timeout error between the application and the installation service
 - NameNotFoundException message when deploying an application that contains an EJB module
 - JSP compilation error for incorrectly coded JavaServer Pages
 - Various wsadmin errors when installing from the command line or through a script

© Copyright IBM Corporation 2007

Figure 13-9. Typical application installation errors

WA5711.0

Notes:

These errors are runtime exceptions based on the failure of the installation tool as it attempts to install the application, not on the run time of the application itself. So while the symptom being reported is an application failure, the cause is based on something either missing from the application or incorrect with the deployment environment. Some examples of environmental problems are incorrect permissions, lack of connectivity, or an incorrectly specified path or other variable.

Any exception that occurs will be logged to the SystemErr.log file. Some of these exception or errors will have message identifiers with them. For example, the WASX7015E error running wsadmin command “\$AdminApp installInteractive” or “\$AdminApp install” error message has the identifier WASX7015E. This identifier can be traced through the InfoCenter for its exact meaning by doing the following:

1. Navigate to the information center library at <http://www-306.ibm.com/software/webservers/appserv/was/library/>.
2. Select the appropriate information center for your WebSphere Application Server V6.1 product.

3. Once the information center has loaded, search on the identifier WASX7015E by entering it in the search text field near the top and clicking the **Go** button.
4. Look through the search result that best fits your query. In the case of this identifier, it is the 100% result page. The identifier is highlighted where ever it is found in the results.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Typical application start-up errors

- The installation process may be successful but the first attempt to start the application might result in failure
- Some of the errors that might occur when attempting to start the application for the first time could be:
 - *java.lang.ClassNotFoundException*: classname Bean_AdderServiceHome_04f0e027Bean
 - *ConnectionFactory E J2CA0102E*: Invalid EJB component: Cannot use an EJB module with version 1.1 using The Relational Resource Adapter
 - *NMSV0605E*: A Reference object looked up from the context... was sent to the JNDI Naming Manager and an exception resulted.
 - Deployed application that uses Java Native Interface (JNI) code does not start
 - Other name server ("NMSV...") errors

© Copyright IBM Corporation 2007

Figure 13-10. Typical application start-up errors

WA5711.0

Notes:

After an application is installed for the first time its initial state is *stopped*. An explicit start is required in order to bring the application on line. Depending on the environment topology, such as whether the application was deployed into a cluster, the actual *start* operation could vary. There could be a single instance of the application, or many instances that start all at once, or a ripple start where the application start operation is staggered throughout the cluster.

An error during the application start operation is of the *configuration* type. What this means is that the actual installation and deployment was satisfied but now that the runtime bindings are being exercised for the first time, some reference is not satisfied. You can see this by the types of errors that are detected such as Class loader exceptions, versioning problem, naming reference issue, and so on.

The syntax of the error message identifier is broken down as follows:

Syntax: CCCC1234X

where:

cccc

is a four character alphabetic component or application identifier.

1234

is a four character numeric identifier used to identify the specific message for that component.

x

is an optional alphabetic severity indicator. (I=Informational, W=Warning, E=Error)

Instructor notes:

Purpose —

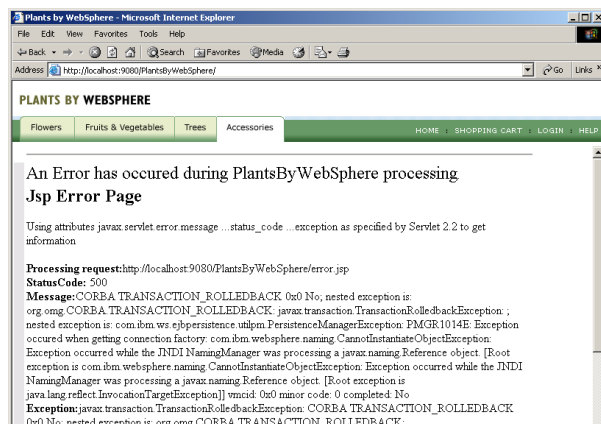
Details —

Additional information —

Transition statement —

Typical application first run errors

- Running the first transaction through the application could result in issues such as:
 - *javax.naming.NameNotFoundException* for unresolved class path problems
 - DNS errors when trying to resolve resources
 - Scoping issues



© Copyright IBM Corporation 2007

Figure 13-11. Typical application first run errors

WA5711.0

Notes:

Problems such as these are environmental problems that affect the operation of the application. The application might be integral, but the environment that it is operating in has unsatisfied references which cause the transaction to fail. The actual symptoms of an environmental problem might be obvious, such as missing content or an error page returned to the browser. However, transaction failure can be more subtle. Your initial request could result in content or results being returned but there might be some graphic missing because a relative path is incorrect or a JSP that has to be created on every request has an error that prevents compilation. Then there are the various browser differences that have to be considered. Depending on the client accessing the content, the type of client being used could have a significant impact on what is considered correct operation. Having your application work perfectly with Firefox but not with other browsers will quickly result in a defect being reported by your users.

Instructor notes:

Purpose —

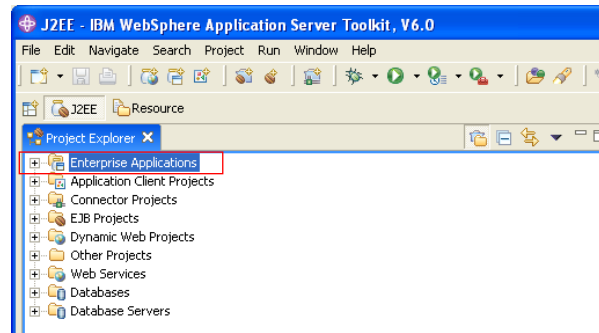
Details —

Additional information —

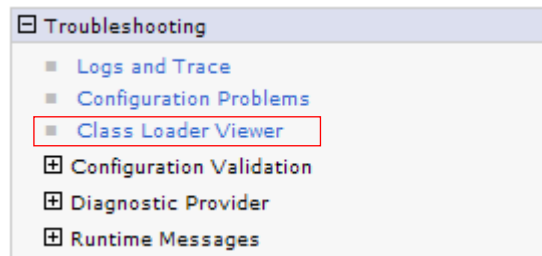
Transition statement —

Application deployment troubleshooting tools

- Application Server Toolkit (AST) provides basic assembly and deployment tooling



- The Class Loader Viewer
 - Embedded in the **Troubleshooting** section of WebSphere Application Server V6.1



© Copyright IBM Corporation 2007

Figure 13-12. Application deployment troubleshooting tools

WA5711.0

Notes:

The *Application Assembly Toolkit* (AST) is an Eclipse-based tool that provides a rich integrated development interface (IDE) for developing and deploying J2EE applications. All of the functionality of the AST is present in the other IBM Eclipse-based products such as WebSphere Studio Application Developer, Rational Application Developer, and WebSphere Studio Enterprise Developer. Therefore, any information about AST in this topic concerning troubleshooting applications can also be applied to these products.

Resolving an application deployment issue might require changes to the application, the environment, the configuration, or all 3.

Making changes to the application can be accomplished using the AST or one of the other Eclipse-based IDEs from IBM. If there is a JSP compilation problem or a configuration change then the AST provides facilities for making changes to the code and the application deployment information.

Class loading is a critical part of the application server environment. The order or precedence of the class loaders are responsible for co-existence between the services provided by WebSphere Application Server V6.1 and the application functionality. As of

version 6.0.2 of the WebSphere Application Server the *Class Loader Viewer* provides visibility into the relationship between the class loader, the class path, and the class. The *drill down* capabilities that the *Class Loader View* provides greatly facilitates resolution of class loader problems.

The WebSphere Application Server V6.1 deployment environment and the application deployment descriptors are tightly linked and require correct information in order for the application to run successfully. The WebSphere Application Server V6.1 environment provides the services that the application leverages through namespace references, such as JNDI names for data sources and resource connections in general. The administrative console installation wizard and the *wsadmin* script interpreter provide APIs for specifying the bindings between the application and the WebSphere Application Server V6.1 containers.

Instructor notes:

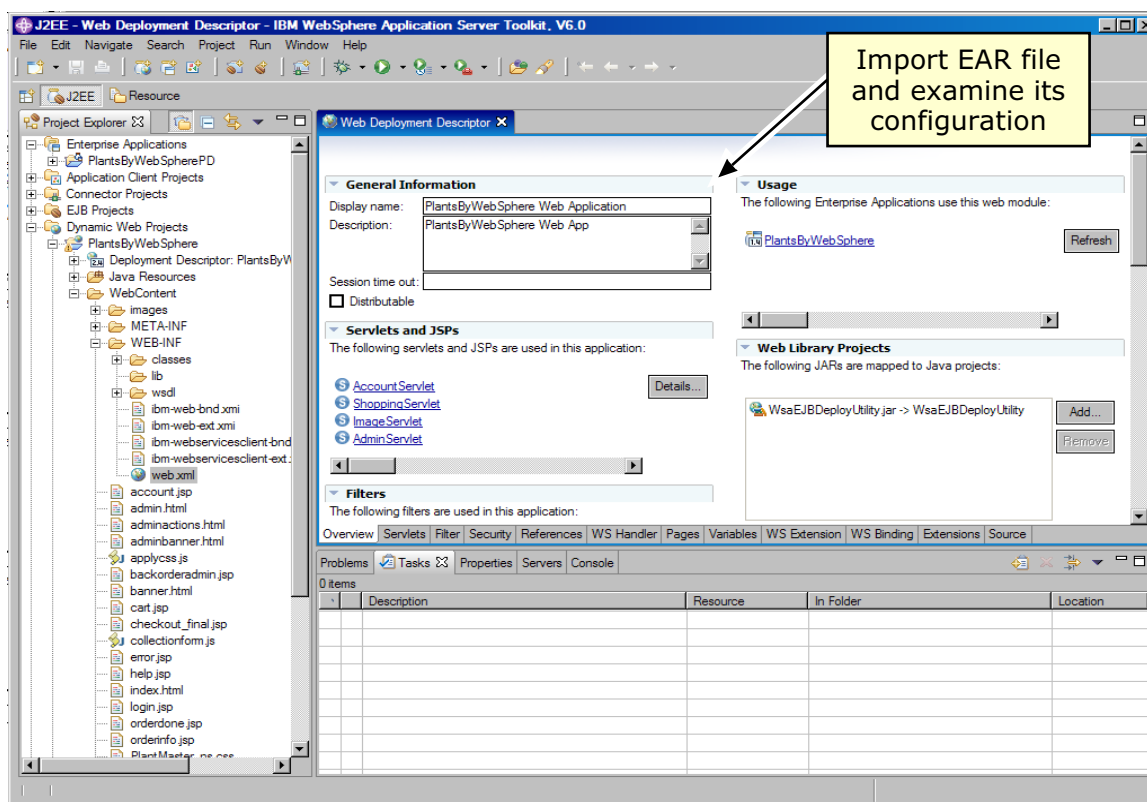
Purpose —

Details —

Additional information —

Transition statement —

WebSphere Application Server Toolkit



© Copyright IBM Corporation 2007

Figure 13-13. WebSphere Application Server Toolkit

WA5711.0

Notes:

Using the AST requires some application development knowledge and skills. The AST displays projects based on *perspectives*, and a brief description of the J2EE perspective follows.

The J2EE perspective breaks the enterprise application down into categories such as *Enterprise Application*, *EJB Projects*, and *Dynamic Web Projects*. The *Enterprise Application* category lists all of the enterprise applications currently loaded into the AST. The EJB and Web modules that are part of the enterprise application are listed under the *EJB Projects* and the *Dynamic Web Projects* categories, respectively.

The enterprise application and its modules contain deployment descriptors that are specific to that component. The deployment descriptor for the *Enterprise Application* lists each of the modules that make up the application and any application level security descriptors. The deployment descriptor for the *EJB Project* contains enterprise bean, session, database and security information, and anything else that the EJBs need for integration with the EJB container. The *Dynamic Web Project* deployment descriptor contains information pertaining to servlets, JSPs, HTML files, and security information.

Instructor notes:

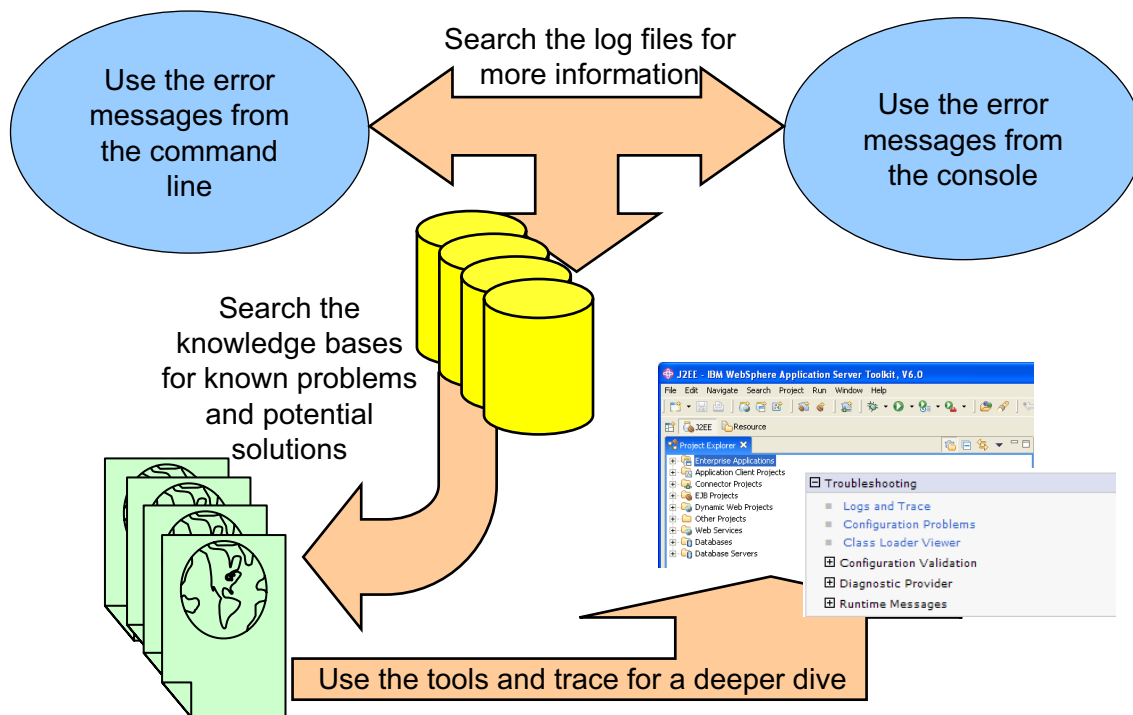
Purpose —

Details —

Additional information —

Transition statement —

Resolving an application installation error



© Copyright IBM Corporation 2007

Figure 13-14. Resolving an application installation error

WA5711.0

Notes:

Some problems that can be resolved using the AST:

- "Timeout!!!" error displays when attempting to install an enterprise application in the administrative console

If this problem occurs, open the enterprise application in the AST, right-click the enterprise application, and select **Deploy** from the pop-up menu. This action creates a file with a name like `Deployed_file_name.ear`. Then, in the administrative console, install the deployed.ear file.

- `NameNotFoundException` message displays when deploying an application that contains an EJB module

To resolve this problem, open the enterprise application in the AST and expand the `ejbModule` directory of the Enterprise JavaBean (EJB) project. Either remove the source files or include all dependent classes and resource files on the class path.

- SRVE0026E: [Servlet Error] - [Unable to compile class for JSP file

This type of problem is a code defect. The AST can be used to edit the incorrect JSP file and verify the changes. Highlight the JSP in the Project Explorer view, right-click it and select **Run Validation** from the pop-up menu.

Resolving application deployment issues using the AST will usually involve changing the information in one or more of these deployment descriptors, saving the changes, and exporting the enterprise application for deployment to the application server.



Note

About Timeout!!!: By loading the application into the AST and exporting it with a new name such as `Deployed_file_name.ear`, the timeout will cease to be a factor. The timeout occurs because the application has not been deployed.

- Data definition language (DDL) generated by an assembly tool throws SQL error on target platform

If you receive SQL errors in attempting to execute data definition language (DDL) statements generated by an assembly tool on a different platform, for example if you are deploying a container-managed persistence (CMP) enterprise bean designed on Windows onto a UNIX operating system server, try the following actions,

Browse the DDL statements for dependencies on specific user identifiers, passwords, specific server names and correct as necessary.

Refer to the message reference of the vendor for causes and suggested actions regarding specific SQL errors.

Sometimes, Ctrl-M characters gets appended to the DDL files when they are moved from Windows to UNIX platforms, which will eventually cause errors while executing these DDL files on UNIX platforms (Edit the DDL file in a text editor and remove extra characters to rectify this).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting class loader problems

After completing this topic, you should be able to:

- Describe the class loading delegation model
- Describe the phases of class loading
- Configure appropriate class loader modes and policies
- Troubleshoot class loader problems using the Class Loader View
- Recognize common class loader exceptions
- Resolve ClassNotFoundException problems

© Copyright IBM Corporation 2007

Figure 13-15. Troubleshooting class loader problems

WA5711.0

Notes:

Instructor notes:

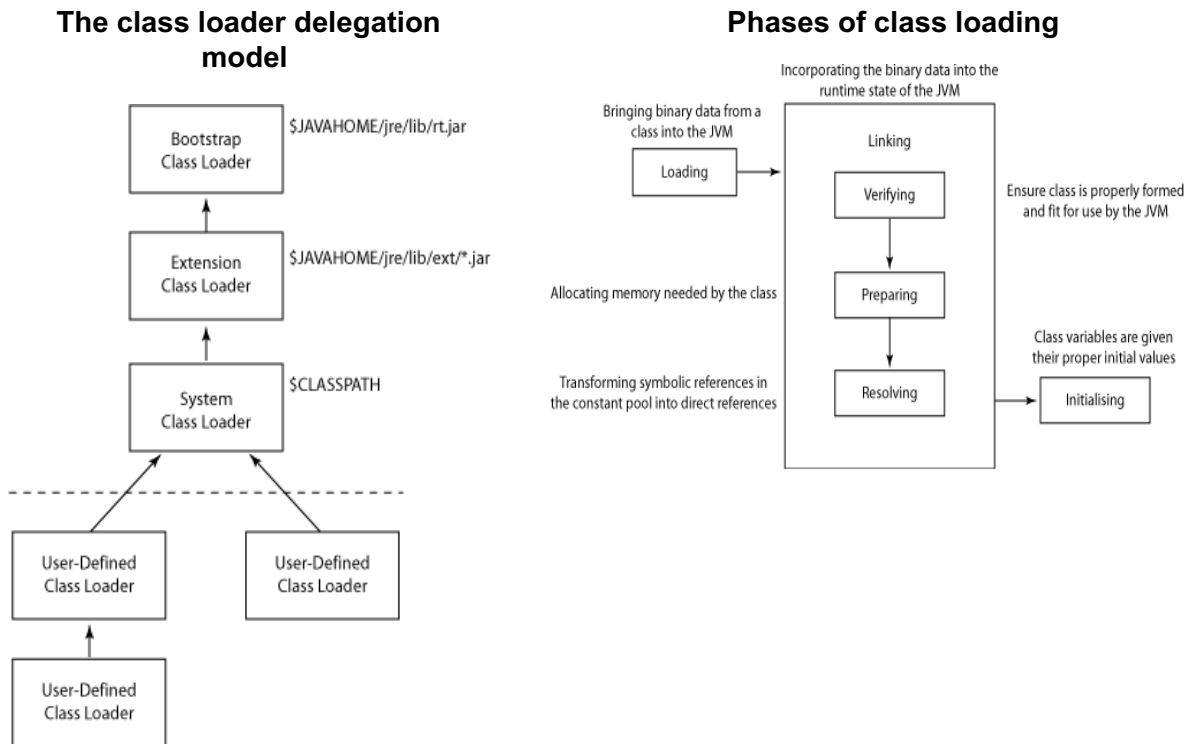
Purpose —

Details —

Additional information —

Transition statement —

Class loading concepts



© Copyright IBM Corporation 2007

Figure 13-16. Class loading concepts

WA5711.0

Notes:

The hierarchy of class loaders used by WebSphere from highest to lowest are:

- Java class loaders
- WebSphere extension class loaders
- Application module class loader
- Web module class loader

Class loading is controlled by three parameters:

- Class loader mode
- War class loader policy
- Isolation policy

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Class loader modes and policies

- Class loader mode
 - **PARENT_FIRST**(default): The class loader delegates the loading of classes to its parent classloader before attempting to load the class from its local class path
 - **PARENT_LAST**: Class loader attempts to load classes from its local class path before delegating the class loading to its parent
 - Allows overriding of higher level classes by searching locally first
- war class loader policies
 - **Module** (default): Web module class loader will be used to load classes
 - **Application**: Application module class loader loads the Web module, resulting in the sharing of classes between modules
- Class loader isolation policies
 - **Single**: Applications are not isolated
 - **Multiple**: Applications are isolated from each other

© Copyright IBM Corporation 2007

Figure 13-17. Class loader modes and policies

WA5711.0

Notes:

The WebSphere Application Server V6.1 information center contains excellent information regarding class loading.

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/crun_classload.html

Instructor notes:

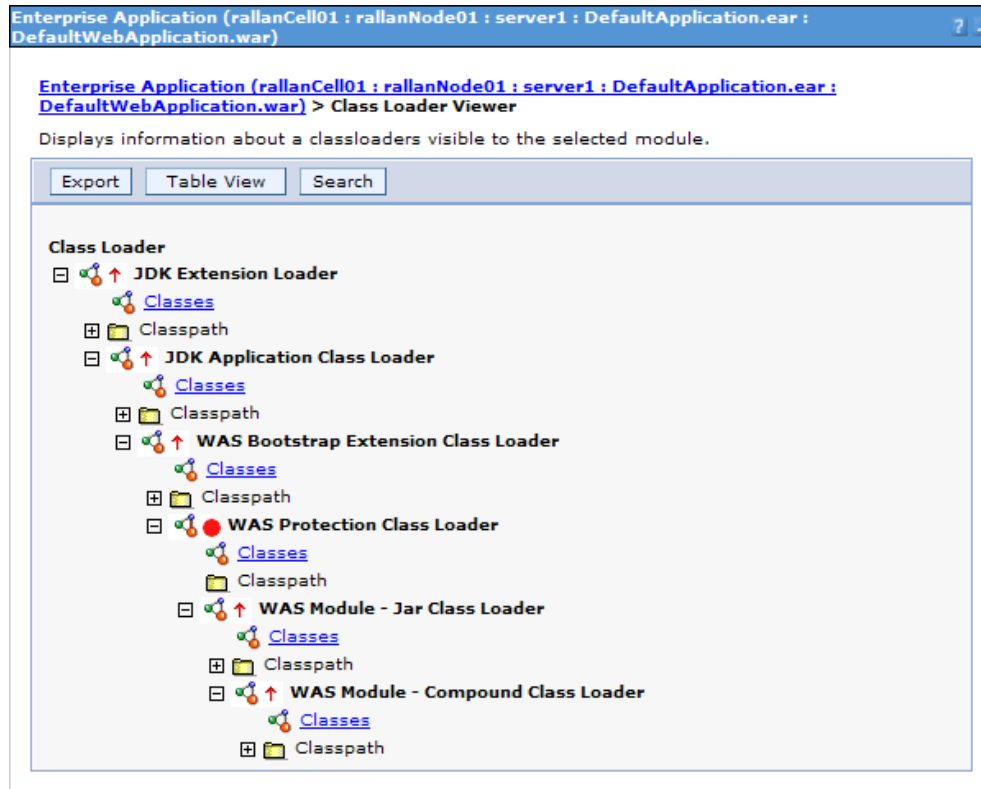
Purpose —

Details —

Additional information —

Transition statement —

Class Loader Viewer example



© Copyright IBM Corporation 2007

Figure 13-18. Class Loader Viewer example

WA5711.0

Notes:

The image, above, shows an expansion of the class loaders for a Web application module. At the top level is the *JDK Extension Loader*, followed by the *JDK Application Class Loader*. By having the *JDK Extension Loader* at the top level, the *JDK* can be extended by deploying new functionality to the directory specified by the *java.ext.dirs* variable. The same reasoning is used for the *WebSphere Application Server V6.1* class loading by having the *WebSphere Application Server V6.1 Bootstrap Extension Class Loader* take precedence over the *WebSphere Application Server V6.1 Protection Class Loader*: The application server can be extended by deploying new functionality to the directory specified by the *ws.ext.dirs* variable.

The *WebSphere Application Server V6 Module – Jar Class Loader* is responsible for loading the *EJB jar* for this particular application. The *WebSphere Application Server V6.1 Compound Class Loader* loads the classes and resources of the enterprise application and the Web module. There will be one of these class loaders for each application.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Class loading problems (1 of 2)

- Common exceptions that can occur as the result of a class loader problem
 - `ClassCastException` – This exception signifies that either the class referenced or the class loader that attempted to load the class are of different types
 - `ClassNotFoundException` – This exception signifies that either the class is not visible, a class it references is not visible, or the class loader API is being used incorrectly
 - `NoClassDefFoundException` – This exception signifies that the class is not in the logical class path
 - `UnsatisfiedLinkError` – This error occurs because a native library is not available or cannot be found

© Copyright IBM Corporation 2007

Figure 13-19. Class loading problems (1 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Class loading problems (2 of 2)

- A class loader problem will not be detected until the first time the class is required and the JVM attempts to load it
- Use Class Loader Viewer utility in the administrative console to help resolve the problem
 - From the administrative console select **Troubleshooting -> Class Loader Viewer**
- The Class Loader Viewer is structured as a hierarchy based on
 - Cell
 - Nodes in the cell
 - Servers in each node
 - Application on each server
 - Each module in the application
- Each module is represented as a hyperlink that leads to a page containing the class loader hierarchy
 - Each class loader in this view contains a tabular view of the class path that it follows and the classes that it loads to satisfy the dependencies of the module

© Copyright IBM Corporation 2007

Figure 13-20. Class loading problems (2 of 2)

WA5711.0

Notes:

The hierarchy of class loaders used by WebSphere Application Server V6.1, from highest to lowest, are:

1. Java class loaders
2. WebSphere extension class loaders
3. Application module class loader
4. Web module class loader

Class loading of WebSphere Application Server V6.1 enterprise applications are controlled by three parameters: The class loader mode, the war class loader policy and the Isolation policy.

The class loader mode parameter can be specified as *parent first* or *parent last*. The default in WebSphere Application Server V6.1, *parent first*, causes the class loader to delegate the loading of classes to its parent class loader before attempting to load the class from its local class path while *parent last* causes classes to be loaded from the current

class path, first. *Parent last* is useful to override functionality of a higher scope and leverage functionality that is local to the application, instead.

The *war class loader policy* can be specified as *module* or *application*. The default setting is *module*, meaning that the Web module class loader will be used to load classes for the Web module. Changing the setting to *application* has the effect of using the application module class loader to load the Web module, resulting in the sharing of classes between modules of the same application. If the isolation policy is set to *Single*, then all applications share the same class loader resulting in resources being shared globally among the applications.

The *Isolation policy* has 2 values: *Single* and *Multiple*. The application class-loader policy controls the isolation of applications that are running in the system. When set to *Single*, applications are not isolated. When set to *Multiple*, applications are isolated from each other.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

How to resolve a ClassNotFoundException problem

- Ask yourself the following questions
 - What file cannot be found?
 - Where is the file that cannot be found?
 - Where is the file that is looking for it?
 - Does it have visibility to the file it is trying to load?
 - Which class loader is attempting to load the class?

- If you answer these questions you should be able to correct a ClassNotFoundException problem by enabling visibility to the class that cannot be found

© Copyright IBM Corporation 2007

Figure 13-21. How to resolve a ClassNotFoundException problem

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Gathering information for deployment problems

- From the administration console, enable diagnostic trace on the specific application server using the following trace string.
 - `*=info:com.ibm.ws.management.*=all:com.ibm.websphere.management.*=all`
- If the deployment is done using wsadmin, enable wsadmin tracing.
 - Edit wsadmin.properties file located under `<profile_root>/properties` folder.
 - Uncomment the following line:
 - `#com.ibm.ws.scripting.traceString=com.ibm.*=all=enabled`

© Copyright IBM Corporation 2007

Figure 13-22. Gathering information for deployment problems

WA5711.0

Notes:

1. In the administrative console, navigate to **Troubleshooting -> Logs and Trace -> dmgr -> Diagnostic Trace -> Change Log Detail Levels**.



Note

If you are in a Base environment, select **server1** instead of the **dmgr** server.

2. Select the **Runtime** tab.
3. In the **Change Log Detail Levels**, modify existing trace string to:


```
*=info:com.ibm.ws.management.*=all:com.ibm.websphere.management.*=all
```

Gather the collector tool before submitting the information to IBM Technical support. The collector tool is available under `<WAS_HOME>/bin` and it should be run from a temporary directory created outside of `<WAS_HOME>` directory.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. What are some of the common problems that can occur when deploying an application?
2. Name three deployment descriptors you might encounter in an enterprise application.
3. What are the two parameters that control the class loading in WebSphere Application Server V6.1?
4. What are the three stages where problems could be encountered while deploying an application to WebSphere Application Server V6.1?

© Copyright IBM Corporation 2007

Figure 13-23. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1.
 - a.
 - b.
 - c.
 - d.
 - e.
2.
 - a.
 - b.
 - c.

3.

a.

b.

4.

a.

b.

c.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions (1 of 2)

1. What are some of the common problems that can occur when deploying an application?
 - a. **Application is not visible.**
 - b. **Any validation error**
 - c. **The application installation process times out.**
 - d. **A *NameNotFoundException* occurs.**
 - e. **The newly deployed application does not respond to a request.**

2. Name three deployment descriptors you might encounter in an enterprise application.
 - a. **The Enterprise Application Deployment Descriptor**
 - b. **The EJB Deployment Descriptor**
 - c. **The Dynamic Web Module Deployment Descriptor**

© Copyright IBM Corporation 2007

Figure 13-24. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions (2 of 2)

3. What are the two parameters that control the class loading in WebSphere Application Server V6.1?
 - a. **Class loader mode**
 - b. **Class loader Isolation policies**

4. What are the three stages where problems could be encountered while deploying an application to WebSphere Application Server V6.1?
 - a. **While installing, deploying, or updating the applications**
 - b. **When starting the application for the first time**
 - c. **When running the application for the first time**

© Copyright IBM Corporation 2007

Figure 13-25. Checkpoint solutions (2 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Detect an application deployment issue in WebSphere Application Server V6.1
- Determine the root cause of an application deployment problem by using the WebSphere Application Server V6.1 log files
- Resolve the application deployment issue by using the administrative console and the Application Server Toolkit (AST)
- Enable on WebSphere servers and wsadmin during deployment process in order to debug a deployment problem
- Identify and resolve class loader issues using Class Loader Viewer

© Copyright IBM Corporation 2007

Figure 13-26. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 8: Troubleshooting application problems

© Copyright IBM Corporation 2007

Figure 13-27. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 14. Server start failures

Estimated time

00:30

What this unit is about

Application server start failure occurs because of system state, security, connectivity, or configuration problems. Such problems could have been introduced during an application deployment, environment change, system upgrade, or by any other activity that alters the system or its environment. Regardless of why or how it occurs, a failure during the start of an application server or one of its components threatens the successfulness of any project and requires immediate attention. This unit looks at the symptoms, the most common causes, and the resolution of application server start failures.

What you should be able to do

After completing this unit, you should be able to:

- Detect a server start failure
- Determine the root cause of the application server start failure
- Resolve the server start failure issue

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

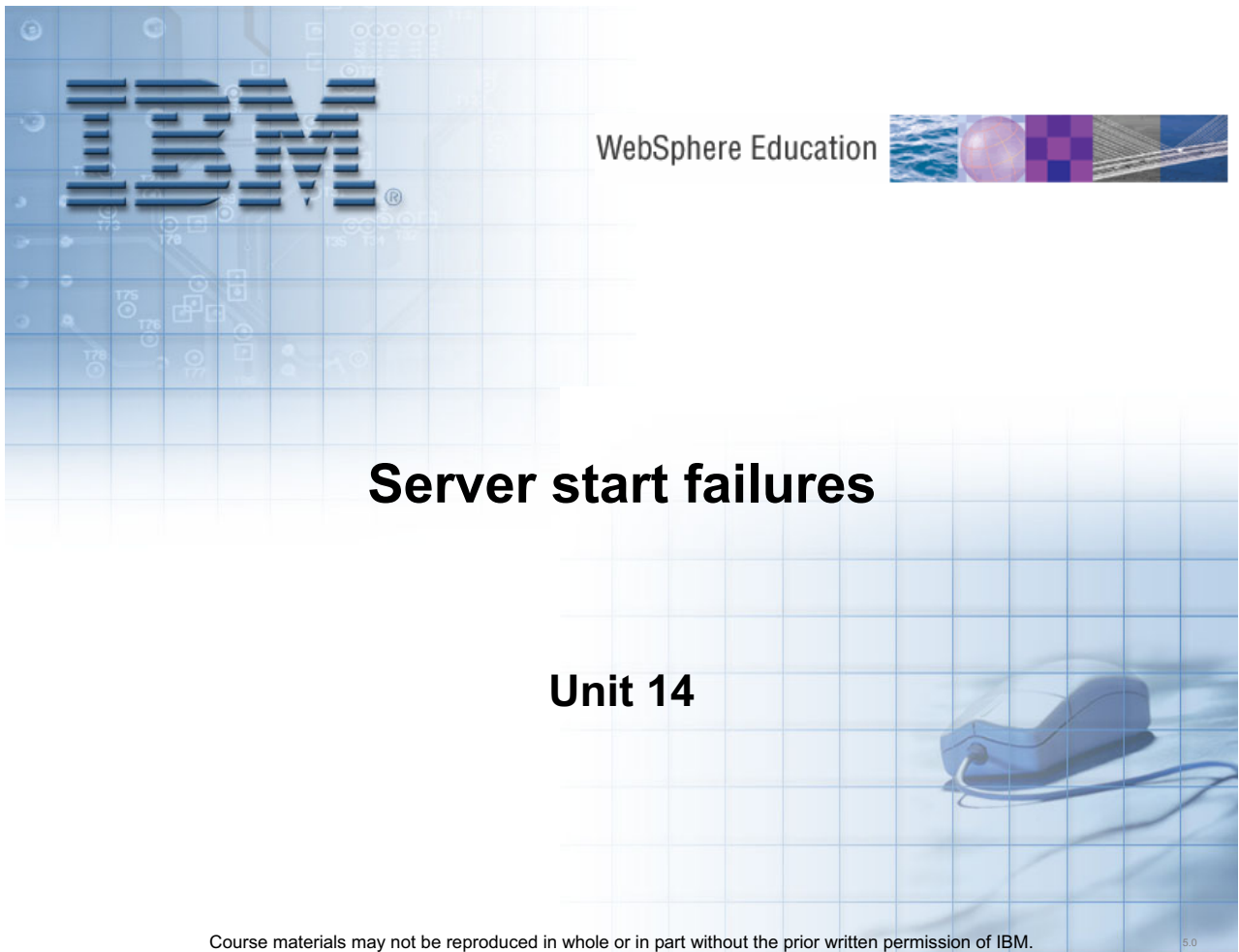
References

WebSphere Application Server, version 6.1 information center

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp> Server startup problems

IBM Troubleshooting Guide for WebSphere Application Server

<http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg27005324>



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.0

Figure 14-1. Server start failures

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe the server start process
- Detect a server start failure
- Determine the root cause of the server start failure
- Resolve the server start failure issue
- Resolve a server stop failure

Figure 14-2. Unit objectives

WA5711.0

Notes:

Throughout the life of an enterprise environment, various activities will take place such as topology changes or maintenance upgrades to the middleware or middleware components. Additionally, new applications and users will be added to the system and obsolete applications and inactive user accounts will be removed. A WebSphere Application Server V6.1 topology change will occur when a node is added or removed, a cluster is created, or a cell is defined. Maintenance patches to the application server are common and will be applied at various times into the production, maintenance, test, or development environment. Any of these activities could alter the runtime environment in a negative manner, resulting in an application server start failure. The ability to detect, determine, and resolve the issue is extremely important to the success of the business that leverages the environment. Therefore, this unit investigates some of the common causes for application server start failure and demonstrates how to resolve such an issue in a timely manner.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

What happens when a server starts

- Issue the command: `startServer server1`
- Two JVMs are actually launched (except on iSeries)
- The first JVM is the Systems Management server launch utility
 - Locates and reads the appropriate configuration (`server.xml`)
 - Spawns the second JVM, which is the actual server process
- The server launching process waits until it receives status back from the server process, then exits
 - Unless you specified the **-nowait** parameter

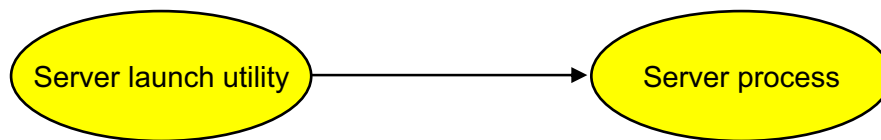


Figure 14-3. What happens when a server starts

WA5711.0

Notes:

For the OS/400 (i5/OS) platform, there actually is *one* JVM. The OS/400 version of `startServer` calls a native program that spawns off another native program that hosts the application server JVM. This was done mainly for performance reasons—having two JVMs resulted in long startup times that was not acceptable on OS/400. The native program performs some process setup and performs the same function as Systems Management server launch utility as far as collecting all information needed by the application server, except it does so natively, then creates a JVM and invokes the application server code using JNI.

Spawning is achieved by constructing a command (`java -Dxxx -classpath x;y;z` and so on) and executing it.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Server start messages

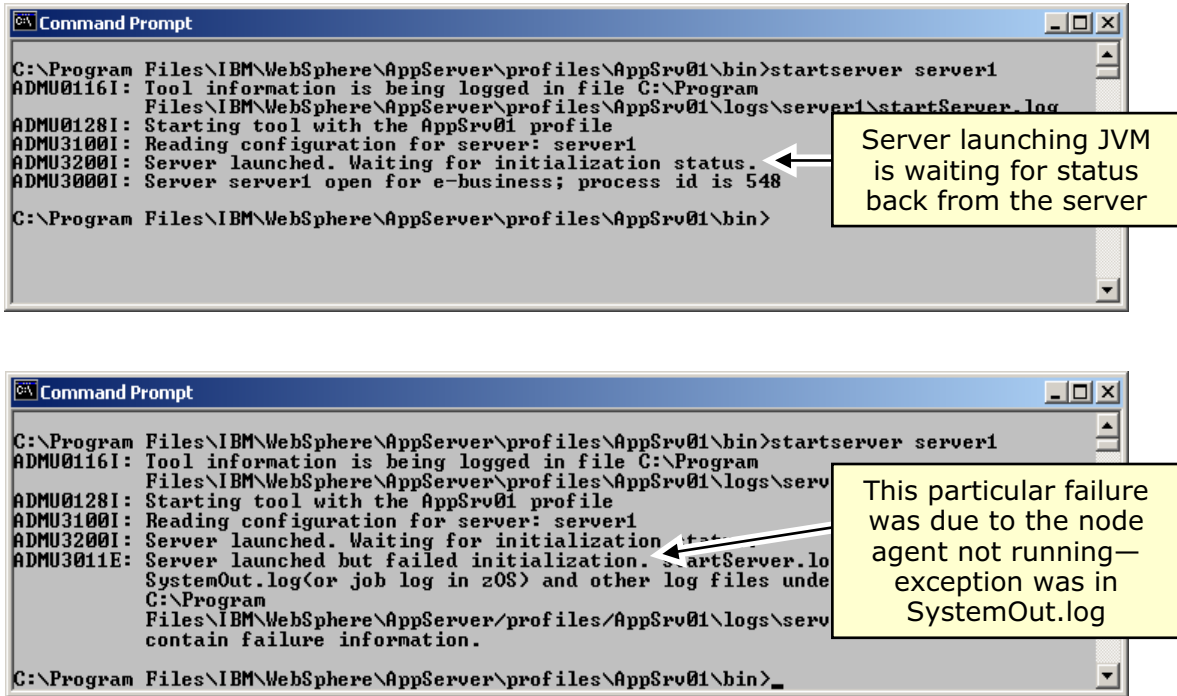


Figure 14-4. Server start messages

WA5711.0

Notes:

In the successful server start show above, notice that the console messages (ADMU) are from the launch utility and the process ID for the actual JVM of server1 is reported.

The start failure shown above was caused by the fact that the node agent was not running. This condition is clearly reported in the SystemOut.log and SystemErr.log files of server1.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Server boot up process

- A relatively small amount of WebSphere code is placed onto the class path of the JVM. This includes the bootstrap code.
- The bootstrap code establishes a class loading environment for the rest of the install image.
 - In V6 and earlier, this meant constructing a class loader that picked up everything in the lib directory (and others).
 - In V6.1 and beyond, this means establishing a proper OSGi environment so that the bundles are loaded.
- The bootstrap code then branches into the “main” of the server
 - This main performs some preliminary processing (like loading the server.xml document) and then begins starting the various components of the system.

Figure 14-5. Server boot up process

WA5711.0

Notes:

Instructor notes:

Purpose —

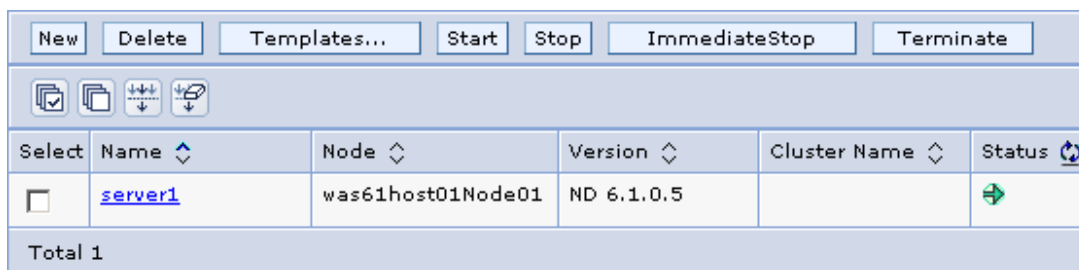
Details —

Additional information —

Transition statement —

Starting a server from the administrative console

- In an ND cell, application servers may be started from the administrative console
- This server start process differs from the command line
 - Does not use a separate launch utility JVM
 - The node agent spawns the server process
 - No data is streamed to the startServer.log




Select	Name	Node	Version	Cluster Name	Status
<input type="checkbox"/>	server1	was61host01Node01	ND 6.1.0.5		
Total 1					

Figure 14-6. Starting a server from the administrative console

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Detecting a failed server start (1 of 2)

- An application server failure during the start process is reported as an error message in the command window or administrative console
- Further details on the error are logged as exceptions in the server logs
 - native_stderr.log
 - native_stdout.log
 - startServer.log
 - SystemErr.log
 - SystemOut.log

} If server fails early, you may *only* see these two logs
- If the server started from the administrative console of the deployment manager, then the errors might be viewable in the Runtime Error message panel
 - Select **Troubleshooting -> Runtime Message -> Error -> Runtime Events**
 - Also check the SystemOut and SystemErr log files of the node agent
- If the application server is started using the wsadmin script interpreter, the startServer.bat (Windows) or the startServer.sh (UNIX) script, the error message will be printed in the command window

Figure 14-7. Detecting a failed server start (1 of 2)

WA5711.0

Notes:

Note on the `<server_name>.pid` file: Many customers assume that if this file is present, the server is definitely running. Sometimes, the file is just left over after the server crashed. Every application server, node agent, and deployment manager will have a `<server_name>.pid` file in its logs directory when the server is running. This file is removed when the server is stopped gracefully using a command or the administrative console.

In a federated or clustered environment, a single application server failure is difficult to detect. Because a server farm is designed for high availability, a malfunctioning application server might not be immediately apparent to users because their requests will continue to be fulfilled, and it might not be detected by administrators unless they are actively viewing the administrative console. Therefore, it is common in large enterprises to employ system monitoring tools that check the integrity of each node in the topology to verify that it is active. When inactivity is sensed by a monitoring tool, an event is sent to the administrator in the form of an e-mail or page.

One example of a systems management tool that can monitor the enterprise for failure is Tivoli Composite Application Management (ITCAM). ITCAM is an enterprise level

monitoring and management application that can create historical reports about the performance of enterprise middleware, send pre-configured events when thresholds are exceeded or rules are violated, and perform systems management tasks across a logical topology.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Detecting a failed server start (2 of 2)

- During the start of an application server, various initialization steps occur
- By looking at the SystemOut.log, the startup process is traced
 - The current environment is initialized, including
 - Host operating system
 - Java virtual machine (JVM)
 - Environment variables that reference the classpath and library locations
 - Trace state
 - Server profile
 - Server configuration
- The message types of interest are prefixed by
 - WSVR (server runtime messages)
 - ADMU (server launch utility messages)
- On a successful application server start, you see the message:
`ADMU3000I: Server <server name> open for e-business; process id is <pid>`
- If the server start fails, the message displayed will depend on the cause of the failure.

Figure 14-8. Detecting a failed server start (2 of 2)

WA5711.0

Notes:

During the start of an application server, various initialization steps occur. By looking at the **SystemOut.log**, located in the application server's **logs** directory, the startup process is traced as:

- The current environment is initialized, including the host operating system, the Java virtual machine (JVM), and the environment variables that reference the class path and library locations
- The trace state, the server profile, the server configuration, and the initialization

The message types of interest are prefixed by WSVR (server runtime messages) and ADMU (server utility messages)

On a successful application server start, the message: `ADMU3000I: Server <server name> open for e-business; process id is <pid>` is displayed. If the server start fails, the message displayed will depend on the cause of the failure.

It is important to point out that a server start failure, when performed from the administrative console, might not create enough trace to detect or determine the type of

error that occurred. As an example, forcing an error by changing the WAS_INSTALL_ROOT and USER_INSTALL_ROOT variables of the server node resulted in no output files being produced and a console message of:

Message Originator: com.ibm.ws.console.core.mbean.MBeanHelper

Message: Could not invoke an operation on object:

WebSphere:platform=common,cell=rallanCell01,version=6.0.2.3,name=NodeAgent,mbeanIdentifier=NodeAgent,type=NodeAgent,node=rallanNode01,process=nodeagent because of an mbean exception:

com.ibm.websphere.management.exception.AdminException: Process failed to launch: server1

This is not enough information to debug the nature of the problem. Therefore, this unit assumes that the **startServer.[bat | sh]** script was used to start the application server in most cases.

Instructor notes:

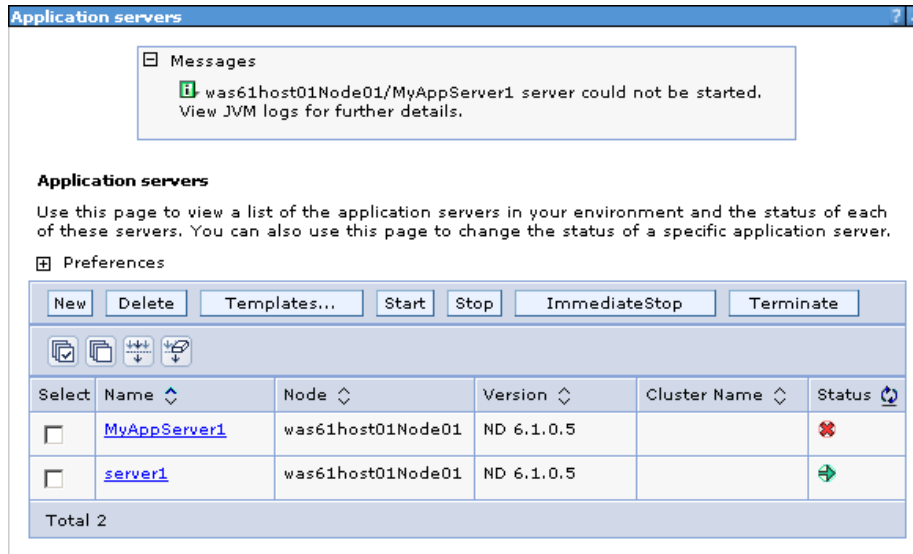
Purpose —

Details —

Additional information —

Transition statement —

Starting a server from the console–failure



In this case no JVM logs were created

Name	Size	Type
MyAppServer1.pid	1 KB	PID File
native_stderr.log	2 KB	Text Document
native_stdout.log	0 KB	Text Document

Figure 14-9. Starting a server from the console–failure

WA5711.0

Notes:

This start failure was caused by setting the min and max heap of the server to 1MB. Notice that only the native_stderr.log and native_stdout.log files were created. In this scenario, the native_stderr.log file shows out-of-memory exceptions while starting the JVM of the server.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Common causes of server start failures

- Port conflict
 - Another process is using the port
 - Server is already started and running
 - Server is already started but JVM is hung or orphaned

- Node agent is not running
 - Application servers must register with the Location Server Daemon which resides in the node agent

- Class loader issue
 - Class path is incorrect or classes are missing

- Out-of-memory issue
 - Not enough native heap
 - JVM max heap size is too small

Figure 14-10. Common causes of server start failures

WA5711.0

Notes:

Prior to WebSphere Application Server V6.1 and server might fail to start because of an authentication failure. Possibly due to an invalid server Principal ID or because the LDAP server could not be reached. In V6.1, exceptions will be thrown on server startup, but the server will start.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Determining server start issues (1 of 3)

- Begin with investigation of the log files to locate any exceptions that have occurred
- Common server start failures will log exceptions
 - Usually, there are multiple exceptions that occur during a server start failure.
 - The initial failure will be the general exception accompanied by a more detailed exception regarding the actual cause of the failure.
- The root cause of the problem will be one or more of the following issues:
 - System State - Orphaned processes may be locking ports
 - Security - Server authentication failures
 - Connectivity - Invalid server ports and host names
 - Configuration - Maximum Java heap too small

Figure 14-11. Determining server start issues (1 of 3)

WA5711.0

Notes:

In this topic, you are going to investigate the various techniques that will help you determine the root cause of the application server start failure. The log files are always the first place to start with a problem like this. Some server start problems are so severe that the Java virtual machine (JVM) will terminate before any information gets put into the SystemOut.log or the SystemErr.log. However, the startServer.log file holds valuable information about the status of the system during the initial phase of activation and should be consulted first. This file is located under the `<WAS_INSTALL_ROOT>/profiles/<profile_name>/logs/<server_name>` directory, along with the SystemOut.log and the SystemErr.log.

Any problem could be a combination of several problems, such as a connectivity or security problem that is caused by an error in the configuration. For example, if the server did not stop properly, leaving it in a bad state, a connectivity issue could arise during restart because a necessary port is currently in use leading to the error message:

An example of a system state problem is reflected in the exception:

```
'ADML0012E: Exception attempting to get free port for status socket {0}
```

Explanation: An exception occurred when attempting to bind to a socket for server status. The socket probably is already in use.

User Response: Record and save the exception information in this message for further problem determination.

```
'ADMU3028I: Conflict detected on port {0}. Likely causes: a) An instance of the server {1} is already running b) some other process is using port {0}
```

Explanation: A port required by this server is already in use by this or another server.

User Response: Do not attempt use a port that is already in use.

```
'ADMU3027E: An instance of the server may already be running: {0}
```

Explanation: An attempt has been made to start a server that is already running.

User Response: Do not attempt to start a server that is already running.

```
'WSVR0009E: Error occurred during startupcom.ibm.ws.exception.RuntimeError: com.ibm.ws.exception.RuntimeError: com.ibm.ejs.EJS
```

Exception: Could not register with Location Service Daemon, which could only reside in the NodeAgent.

Make sure the NodeAgent for this node is up an running.; nested exception is:
org.omg.CORBA.ORBPackage.Invalid

Name: LocationService:org.omg.CORBA.TRANSIENT:

java.net.ConnectException: Connection refused: connect:host=<host>,port=9900 vmcid:
IBM minor code: E02 completed: No'

An example of a configuration problem that leads to a security issue is an improperly specified or missing credentials, leading to an error message such as:

```
'ADMU4113E: Verify that username and password information is on the command line (-username and -password) or in the.client.props file.
```

Explanation: The tool could not contact the remote server because of a credentials problem.

User Response: You may need to pass -username & -password on the command prompt, or update the soap/sas.client.props file.

However, a connectivity issue could also be a result of incorrect configuration, where one component is expecting communication to occur on a port that is actually not correctly specified, resulting in an error such as:

```
'ADMU3040E: Timed out waiting for server initialization: {0} seconds
```

Explanation: The server initialization process has exceeded the timeout limit.

User Response: Perform problem determination on the server to check for possible errors.

```
'ADML0040E: Timed out waiting for server "{0}" initialization: {1} seconds
```

Explanation: The server initialization process has exceeded the timeout limit.

User Response: Perform problem determination on the server to see why it is taking so long to start.

```
'ADML0060W: Cannot contact server "{0}".
```

Explanation: The node agent failed to contact the given server and then assumed the given server is malfunctioning. The given server will be restarted if "autoRestart" attribute is set to "true" in its monitoring policy, which is defined in server.xml file.

User Response: Check SystemOut.log file for this server; Normally the log file shows why the given server could not respond to the node agent. In case "autoRestart" is set to true, check to see whether it is restarted by the node agent.

General configuration problems where path, server, or any other configuration information is improperly specified could result in an error message such as:

```
'ADMU3402E: Server name not specified; must supply the name of a server.
```

Explanation: This tool requires that a server name be provided.

User Response: Specify a server name.

```
'ADMU3522E: No server by this name in the configuration: {0}
```

Explanation: The specified server name cannot be located within the configuration for this node.

User Response: Check that the specified name is spelled correctly and that a directory by that name does exist under the servers directory in the configuration.'

```
'ADML0029E: No configuration defined for server: {0}
```

Explanation: An attempt has been made to start a server that has no configuration.

User Response: Check that the server name entered is not a mistake.

'ADML0062W: Cannot load server.xml for server {0}

Explanation: The node agent failed to load the server.xml file to read the monitoring policy for given server

User Response: Open the problem server.xml to check whether there is any XML syntax error.

'ADML0003E: A configuration error occurred in the ProcessDef Umask property {0}.

Explanation: The value of the Umask property is not in a valid integer format.

User Response: Edit the configuration and change the value of the process Umask property.

'ADML0004E: An exception occurred when attempting to expand variable {0} {1}.

Explanation: The variable in the configuration data cannot be resolved.

User Response: Edit the configuration and change the value of the property to use a variable that can be resolved.'

'ADML0006E: The \${WAS_INSTALL_ROOT} variable is missing.

Explanation: The variable setting for \${WAS_INSTALL_ROOT} value is missing. This variable is required.

User Response: Edit the configuration and provide a \${WAS_INSTALL_ROOT} variable. '

In addition to these errors, error messages with prefix WVRS could also occur. For simplicity, you will concentrate on error messages from the ADMU family during this unit.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Determining server start issues (2 of 3)

- The application server can be started with the trace option specified on the command line
 - `startServer.[bat | sh] <server name> -trace`
 - Start information will stream to the `startServer.log` file (otherwise not available)
- Using the `-trace` option is the same as setting the startup trace state to `com.ibm.*=all`
- The `startServer.log` file is very useful because some failures occur prior to the `SystemOut.log` and `SystemErr.log` trace streams being initialized
 - If the server is started from the administrative console, there is no `startServer.log`. In that case, check the `native_std*.log`, and the node agent logs

Figure 14-12. Determining server start issues (2 of 3)

WA5711.0

Notes:

The `-trace` option during the server start process streams a significant amount of trace information to the **startServer.log** file, beginning with the line:

```
[4/11/06 9:03:43:020 CDT] 0000000a ManagerAdmin I   TRAS0017I: The startup
trace state is *=info:com.ibm.*=all.
[4/11/06 9:03:43:030 CDT] 0000000a AdminTool      3   Our guess at the profile
dir: C:\WebSphere6\AppServer\profiles\AppSrv01
[4/11/06 9:03:43:180 CDT] 0000000a AdminTool      A   ADMU0128I: Starting
tool with the AppSrv01 profile
[4/11/06 9:03:43:180 CDT] 0000000a AdminTool      3   executing utility with
arguments: C:\WebSphere6\AppServer/profiles/AppSrv01/config rallanCell01
rallanNode01 server1 -trace -username <user> -password *****
[4/11/06 9:03:43:180 CDT] 0000000a AdminTool      A   ADMU3100I: Reading
configuration for server: server1
[4/11/06 9:03:44:302 CDT] 0000000a ConfigRootImp < <init> Exit
```

In the case where the server starts but fails initialization, the **startServer.log** contains trace that shows the server starting successfully until the following exception is encountered:

```
[4/12/06 10:56:48:825 CDT] 0000000a ResourceLocat 3
C:\WebSphere6\AppServer\profiles\AppSrv01\config\cells\rallanCell101\clusters\variables.xml not found
  java.io.FileNotFoundException:
  C:\WebSphere6\AppServer\profiles\AppSrv01\config\cells\rallanCell101\clusters\variables.xml (The system cannot find the path specified)
  at java.io.FileInputStream.open(Native Method)
```

The system has been unable to locate a file during initialization time, leading to the initial failure message. It is not clear what impact this failure had on the overall system because it did start without this file.



Note

A cluster was not configured in this test environment. This might be a general failure message that has no adverse affect, at all.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Determining server start issues (3 of 3)

- Enable server verbose modes
 - Verbose class loading
 - Verbose garbage collection

- Can be enabled from the Administrative Console
 - **Application servers -> *server_name* -> Process Definition -> Java Virtual Machine**

- Trace information written to `native_stderr.log` by default

- If Administrative Console is not available, edit the `server.xml` configuration file
 - `verboseModeClass="true"`
 - `verboseModeGarbageCollection="true"`

Figure 14-13. Determining server start issues (3 of 3)

WA5711.0

Notes:

Verbose JNI can also be enabled which provides debug output for native method invocation.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Common configuration error messages

- ADMU3402E: Server name not specified; must supply the name of a server.
- ADMU3522E: No server by this name in the configuration
- ADML0029E: No configuration defined for server
- ADML0062W: Cannot load server.xml for server
- ADML0003E: A configuration error occurred in the ProcessDef Umask property
- ADML0004E: An exception occurred when attempting to expand variable
- ADML0006E: The `#{WAS_INSTALL_ROOT}` variable is missing.

Figure 14-14. Common configuration error messages

WA5711.0

Notes:

Error: ADMU3402E: Server name not specified; must supply the name of a server.

Resolution: The command is ambiguous without a server name specified. This is a common error when starting a server using the **startServer.[bat|sh]** script in a deployment manager environment. Retrieve the server name from the server.xml file or the server name used in the directory path.

Error: ADMU3522E: No server by this name in the configuration: {0} or
ADML0029E: No configuration defined for server: {0}

Resolution: Check the spelling of the server name. If this is not the issue, verify that the server name you are using is the correct server name. With multiple servers, it is easy to lose track of which name applies to which server.

Error: ADML0062W: Cannot load server.xml for server {0}

Resolution: The server.xml file might be corrupted or incomplete. One easy method for checking the validity and format of an XML file is by trying to open it with Internet Explorer (IE). If IE can render it as XML with no errors, then the file is correct. Verify that the server.xml is located in the correct directory. If it is referenced by a symbolic link, check the link for integrity by following it using the **cd** command to change directory. Finally, check the permissions on the server.xml file. Can it be read by the user the application server runs as?

Error: ADML0003E: A configuration error occurred in the ProcessDef Umask property {0}.

Resolution: The permissions under which the Java process executes is incorrect. Therefore, the Java process cannot access file system resources necessary to run the application server. The value needs to be corrected to coincide with the file permissions set for your application server installation. These can be corrected in the administrative console by navigating to the **Servers -> Application Servers -> server_name**. Then, under Server Infrastructure, click **Java and Process Management -> Process Execution**. This panel contains the Umask property.

Error: ADML0004E: An exception occurred when attempting to expand variable {0} {1}.

Resolution: A WebSphere Application Server variable reference includes another variable reference that cannot be expanded because it is incorrectly defined. This error message will specify the actual variable in error. Navigate to the **Environment -> WebSphere Variables** panel and correct the variable.

Error: ADML0006E: The \${WAS_INSTALL_ROOT} variable is missing.

Resolution: Navigate to the Environment > WebSphere Variables and define the WAS_INSTALL_ROOT variable.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Resolving server start issues

- Port conflict
 - Verify server ports using `serverindex.xml` of the node
 - Check for used ports with **netstat -a**, **lsof -i** or port scanning tools
 - Use the **ps -ef** command on UNIX or the **Task Manager** on Windows to determine orphan processes and kill them

- Node agent is not running
 - Start the node agent
 - Check for orphaned node agent process, kill, and restart node agent

- Class loader issue
 - Enable verbose class loading and look for `ClassNotFoundException` exceptions

- Out-of-memory issue
 - Locate Java core dump and analyze with Thread Analyzer tool
 - Locate heap dump and analyze with MDD4J tool

Figure 14-15. Resolving server start issues

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Server stop failures

- Servers fail to respond to a stop command from the administrative console or the command line
- If security is enabled, you must supply both the -user and -password arguments for the following commands:
 - stopServer
 - stopNode
 - stopManager
- If using user ID and password in the soap.client.props file, verify that the login information is valid for the active user registry
- Servers may be hung and will not respond to a stop command
 - Check SystemOut.log files for messages from the ThreadMonitor such as

```
[7/11/07 13:51:54:207 EDT] 0000000d ThreadMonitor W WSVR0605W: Thread
"SoapConnectorThreadPool : 6" (00000032) has been active for 607027 milliseconds
and may be hung. There is/are 10 thread(s) in total in the server that may be hung
```

Figure 14-16. Server stop failures

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Administrative Console stop server options

- In addition to the **Stop** button, the Administrative Console provides
 - ImmediateStop
 - Terminate
- **ImmediateStop**
 - Stops the server
 - Bypasses the normal server quiesce process that supports in-flight requests to complete before shutting down the entire server process
 - Faster than the normal server stop processing, but application clients may receive exceptions
- **Terminate**
 - Deletes the server process that cannot be stopped by the Stop or ImmediateStop commands
 - Application clients may receive exceptions
 - Always attempt an immediate stop before using this option

Figure 14-17. Administrative Console stop server options

WA5711.0

Notes:

These immediate stop is also available when using wsadmin. For example,

Using Jacl: `$AdminControl stopServer serverName immediate`

Using Jython: `AdminControl.stopServer('serverName', immediate)`

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Stop server by killing the java process

- If an application server is hung, you can stop it by killing the Java process
 - First try triggering a thread dump—the javacore file will contain useful diagnostic data
 - You can get the process ID of the server from startServer.log or the `<server_name>.pid` file in the logs directory for the server
 - In a Windows environment, use the Task Manager to stop the task
 - In a UNIX environment, use the `kill -9` command to kill the Java process
- After you have killed the hung Java process, you need to restart the server.
- Stopping a hung process is simply a work-around. The real problem is the hang.
 - Examine any heap dumps and Javacore files

Figure 14-18. Stop server by killing the java process

WA5711.0

Notes:

In the Windows Task Manager, the process ID is not shown by default.

However you can add the process ID (PID) using the **View -> Select Columns** option.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. If the server start fails or partially fails, where might the error messages be streamed?
2. True or False: Server failure will always result in output being available in the SystemErr.log.
3. Name the four categories of issues that might be present during server start failure.
4. What option do you specify during as an argument to the startServer.[bat|sh] in order to gather verbose information during the start process? Where does the information get logged to?

Figure 14-19. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1.
 - a.
 - b.
 - c.
2. T/F? Explain.
3.
 - a.
 - b.
 - c.
 - d.

4.
 - a.
 - b.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

1. If the server start fails or partially fails, where might the error messages be streamed?
 1. **The startServer.log, SystemErr.log, or the SystemOut.log**
 2. **The administrative console**
 3. **The command line console (shell)**
2. True or False: Server failure will always result in output being available in the SystemErr.log.
 - **False. The server start process can fail prior to information being written to the SystemErr.log. This is the purpose of the startServer.log.**
3. Name the four categories of issues that might be present during server start failure.
 1. **System State**
 2. **Security**
 3. **Connectivity**
 4. **Configuration**
4. What option do you specify during as an argument to the startServer.[bat|sh] in order to gather verbose information during the start process? Where does the information get logged to?
 - **The –trace option will force verbose information into the startServer.log file.**

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe the server start process
- Detect a server start failure
- Determine the root cause of the server start failure
- Resolve the server start failure
- Resolve a server stop failure

Figure 14-21. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 9: Troubleshooting server start failures

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 15. Request flow and Web container problems

Estimated time

00:45

What this unit is about

This unit describes the flow of requests from a Web browser to the Web container. Various methods for troubleshooting request flow problems will be discussed.

What you should be able to do

After completing this unit, you should be able to:

- Understand the Web request flow
- Various ways of serving a request
- Enable IHS and Plug-in trace
- Enable application server trace
- Use the Web Container Diagnostic Provider
- Troubleshooting a failing request

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

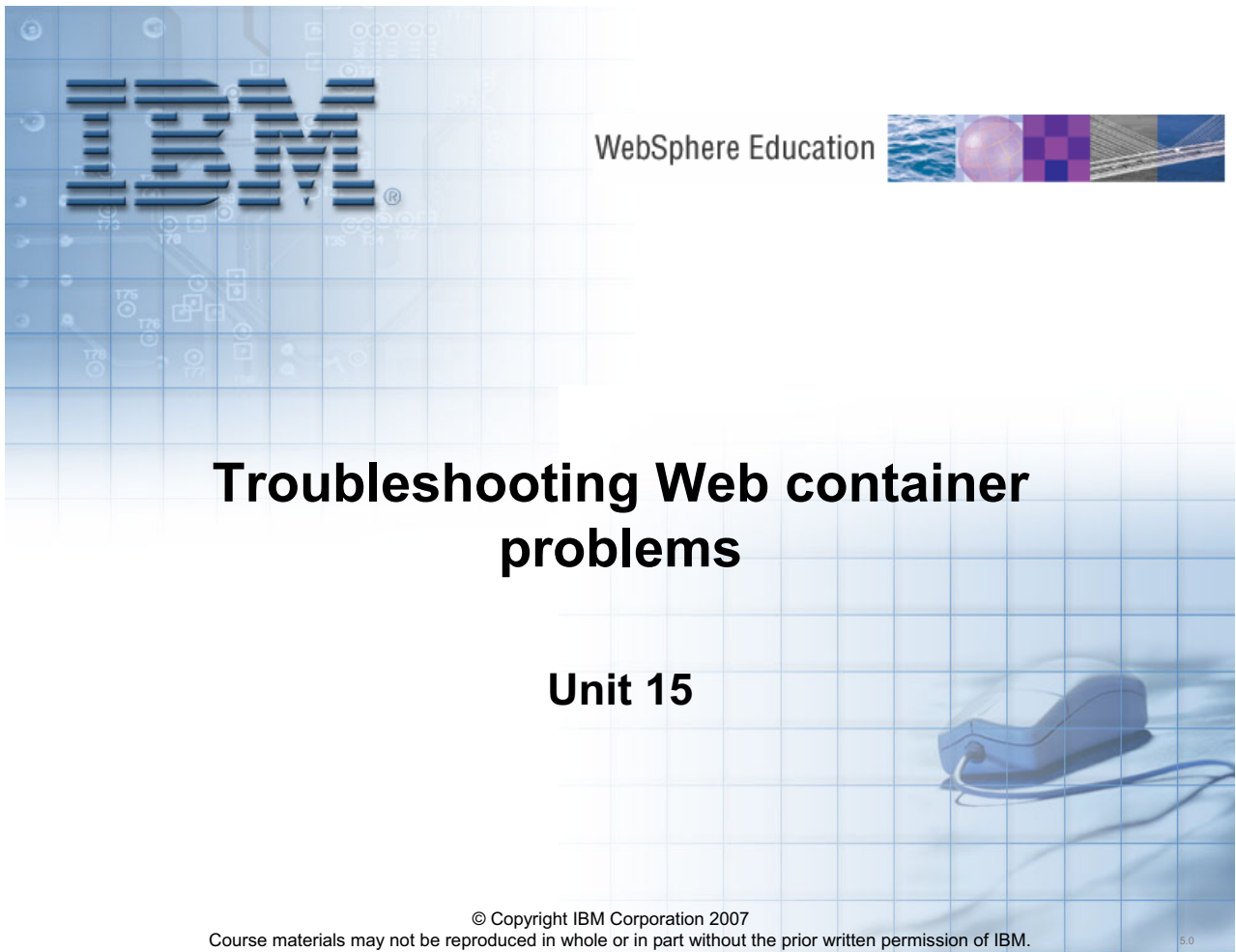


Figure 15-1. Troubleshooting Web container problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe the Web Request flow
- List various ways of serving a request
- Enable IHS and Plug-in trace
- Enable application server trace
- Use the Web Container Diagnostic Provider
- Troubleshoot a failing request

© Copyright IBM Corporation 2007

Figure 15-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose — To introduce the content of this unit.

Details —

Additional information —

Transition statement —

Web request protocol

- Common protocols
 - HTTP
 - HTTPS

- Typical request methods
 - GET
 - POST

- Browser return codes
 - 1xx – Informational (100)
 - 2xx – Successful (200)
 - 3xx – Redirection (302, 304)
 - 4xx – Client Error (400, 404)
 - 5xx – Server Error (500)

© Copyright IBM Corporation 2007

Figure 15-3. Web request protocol

WA5711.0

Notes:

This sheet identifies three areas of interest.

Protocols – This course is limited to these two.

- HTTP is unsecured protocol. Default port for this protocol is 80.
- HTTPS is secured protocol. Default port for this protocol is 443.

Methods – This module will focus on GET and POST. Other methods are HEAD, PUT, DELETE.

- GET request will contain all request information in request headers. Request parameters are encapsulated in URL. There will NOT be any further body attached to GET request.
- POST request will have a POST body followed in addition to the request headers. Content-length header defines the length of POST body. All request parameters will be present in body of the POST request.

Return Codes – These represent the common response codes you will notice in logs when debugging problems

100 – Continued Response

200 – Successful Completion

302 – Temporary Redirection. Response will contain a **Location** header with the redirected location.

304 – Not modified

400 – Bad Request. Client has not formulated the request properly or some network problem.

404 – Not Found. Requested resource is not found in the server.

500 – Server Error. Application Server not available or some runtime error occurred.

Instructor notes:

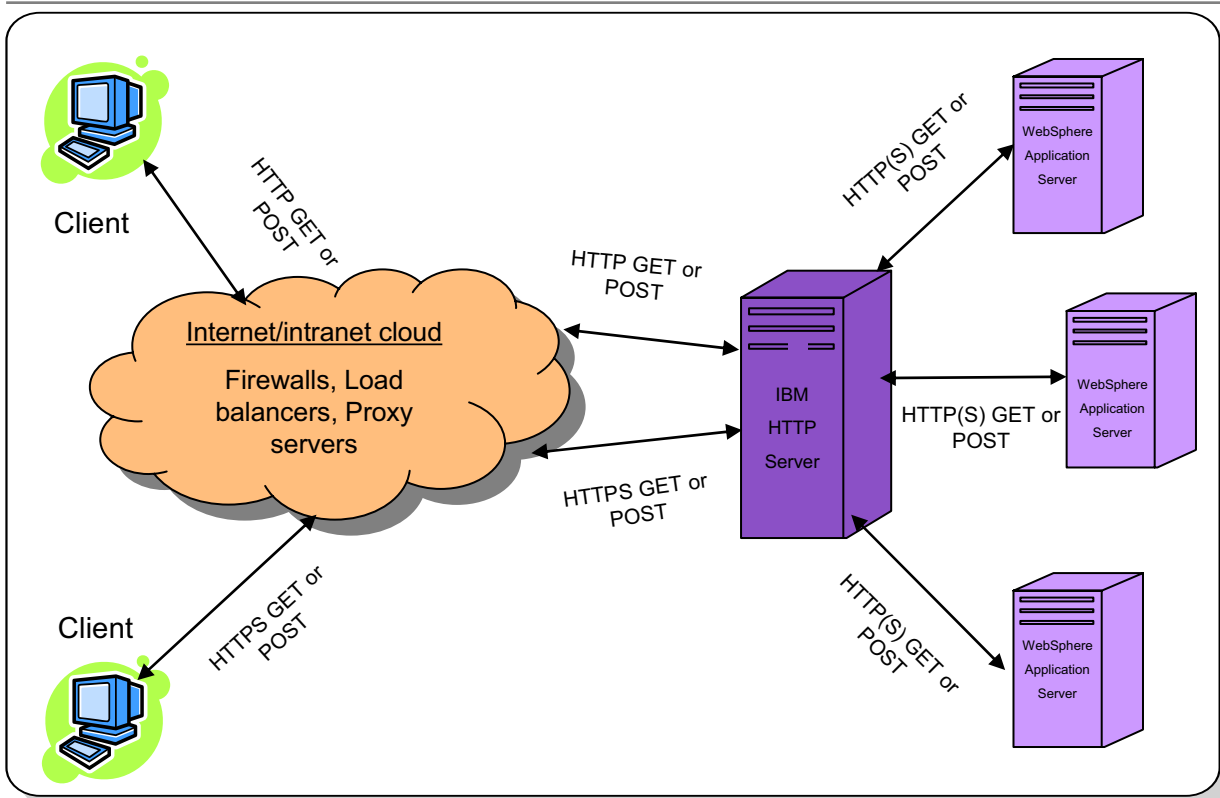
Purpose —

Details —

Additional information —

Transition statement — Before proceeding further on these, take a look at some pictures.

Request flow (HTTP(S))



© Copyright IBM Corporation 2007

Figure 15-4. Request flow (HTTP(S))

WA5711.0

Notes:

This diagram depicts how a request moves from and back to a client. The Internet/intranet cloud graphic depicts possible areas of concern that could also cause request failures. These are not explored in this unit. If you suspect that a request failure is caused by one of these gateways, you can prove or disprove this by taking a packet trace from the IBM HTTP Server box to verify the content and movement of a request. WebSphere Application Server plug-in supports Web Servers like Microsoft IIS, Sun One WebServer, IPlanet and others.

Packet traces are also known as IP traces. These traces record the individual packets from the TCP layer as they move in and out of the problematic box. They are most helpful when you are suspicious that requests are not making it to a problematic box, or that the content of the request doesn't represent what is expected. Reading a packet trace does require an experienced network administrator to interpret the data.

Other tips to expose problems:

- Try pinging the host name to verify that the correct IP of the host box is returned.

- If you have a firewall that blocks ping to the host, you may want to try telnet to Web server port.
- From the Web Server box, bypass the plug-in and attempt a request directly to the application server transport port. You can find the transport port from the Web container transport chain settings of the application server. Assuming application server is listening on 9080 for HTTP traffic, here is an URL:

`http://hostname:9080/myapplication.jsp`

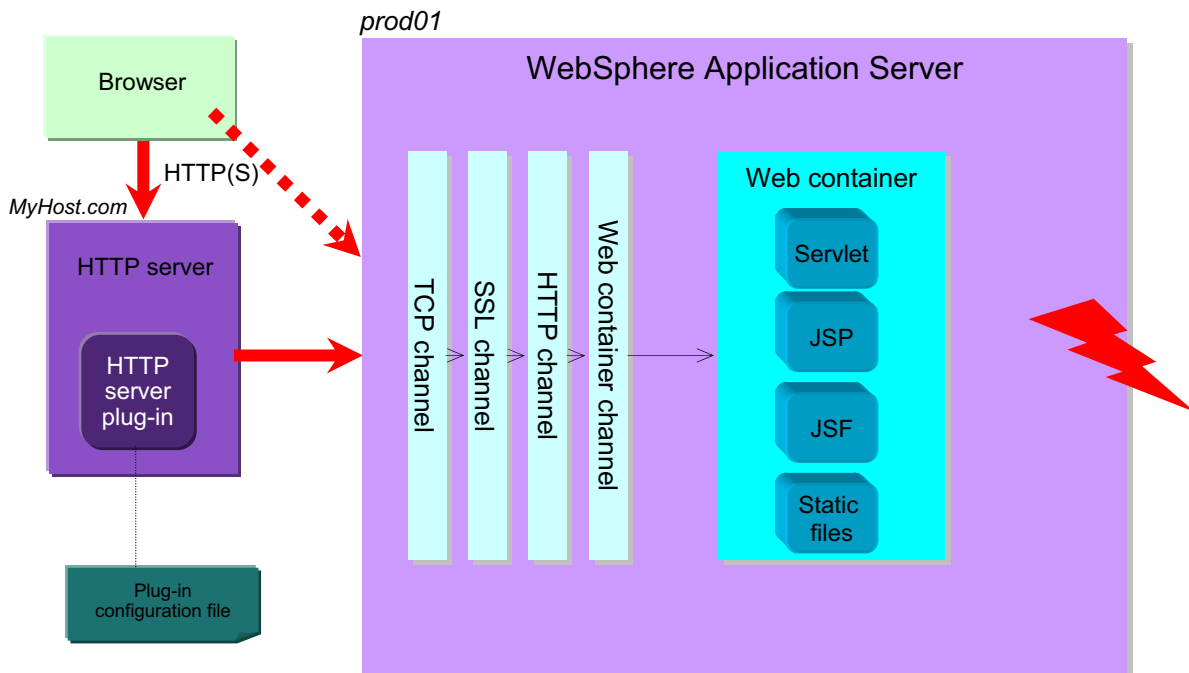
Instructor notes:**Purpose —**

Details — This diagram depicts an architecture that could exist. The cloud is intended to be vague because you do not want to explore these areas directly, but it is worth mentioning to aid in developing the big picture. Also, firewalls can also exist between the Web server and the back-end application servers.

Additional information —

Transition statement — Next slide shows some detailed flow of a request.

Detailed Web request flow



© Copyright IBM Corporation 2007

Figure 15-5. Detailed Web request flow

WA5711.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server showing Web server and key components in WebSphere Application Server that handles request.

The key components in WebSphere Application Server are Channel Framework and Web container. The channel framework replaces the HTTP Transport component in earlier versions (prior to V6).

Channel framework consists of various channels that interact in handling the socket data. A request will go through TCP, SSL (only for HTTPS), HTTP and Web container channels before Web container handles it.

- The TCP channel manages client connections providing asynchronous I/O layer between client and application server threads. The mapping of client connections to threads is generally many to one.
- The HTTP Channel provides a HTTP protocol support for the application server Web serving capabilities.

- The Web container channel provides a dispatching layer between the channel and the servlet container.

There are also some other important components outside of the application server process.

WebSphere Application Server also provides a plug-in for HTTP servers that determines what HTTP traffic is intended to be handled by WebSphere, and routes the requests to the appropriate server. The plug-in is also a critical player in workload management of HTTP requests, as it can distribute the load to multiple application servers, as well as steer traffic away from unavailable servers. It reads the application server configuration from a special XML file, `plug-in-cfg.xml`.

Instructor notes:

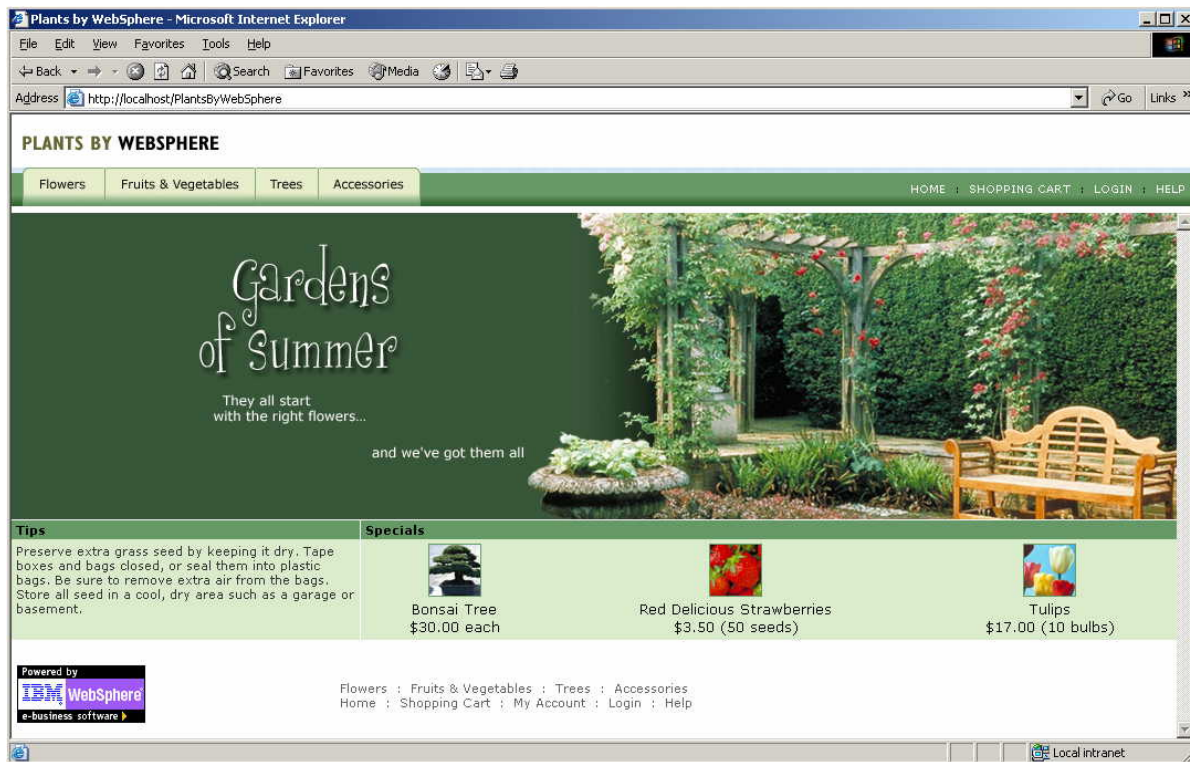
Purpose — To show the runtime flow of a request in WebSphere Application Server.

1. **Details** — The browser is the main interaction mechanism that users will use.
2. A browser communicates with a Web server (HTTP Server). You can also bypass HTTP Server by making a request to servers transport port.
3. The way the request gets into the WebSphere Application Server is from the HTTP Server Plug-in that is loaded with the HTTP Server. This request is forwarded to the transport port within the application server. The channel framework, reads the socket information and provides data to Web container component. Web container will invoke appropriate Web resource based on the request information.

Additional information —

Transition statement —

Typical request methods: GET



© Copyright IBM Corporation 2007

Figure 15-6. Typical request methods: GET

WA5711.0

Notes:

The screen above shows the successful results for a GET request to the PlantsByWebSphere application.

A GET is any request seeking content from a host application.

Instructor notes:

Purpose —

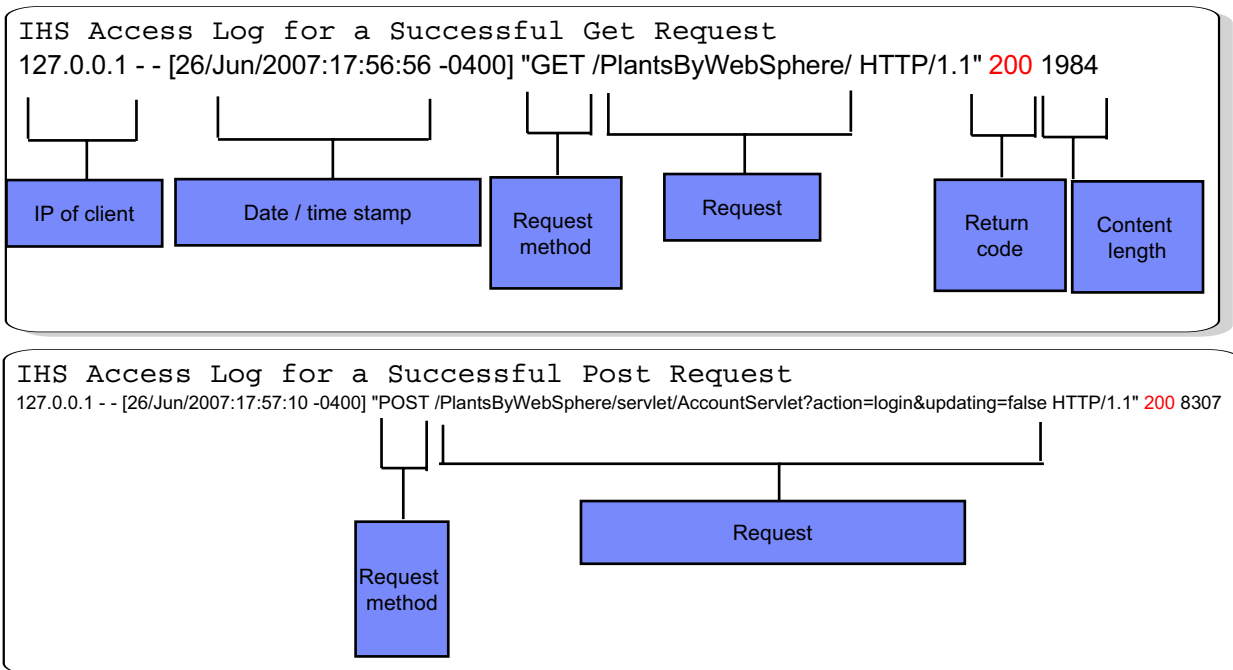
Details — This screen capture is from the PlantsByWebSphere sample application.

Additional information —

Transition statement —

Identifying a working request: Web server access log

- GET and POST requests



© Copyright IBM Corporation 2007

Figure 15-7. Identifying a working request: Web server access log

WA5711.0

Notes:

All requests submitted to the IBM HTTP Server (IHS) are recorded within the Web server access log at the completion of the request. This is where you can verify the response to a specific request. Again, typical working request/response codes are 200, 302, 304.

All responses within the access log are recorded at the “END” of the request cycle. Therefore, if the request fails to be processed on the Web server for some reason, the response may not get posted. And obviously, if a request never makes it to the Web server, it too will not get recorded.

If you want to tack request information you can change the LogLevel to debug in configuration file. This will enable HIS to write detailed information in error.log.

The “-0400” within the access log represents this system time offset from GMT time.

Return codes of 4xx and 5xx are failure codes.

IHS logs are found in the `<IHS_Root>/logs` directory unless otherwise specified within the `httpd.conf` file. The default name for the access log is `access.log`.

Both of the examples above show that a 200 response was returned. The output conforms to the default IHS configuration log format, specified by the following entry in *httpd.conf*:

```
CustomLog logs/access.log common
```

Note that the file name for the access log does not provide the fully qualified path to the file. IHS will assume that this directory exists at the *ServerRoot* as defined within the *httpd.conf* file.

Other values can be added to the output stream by adjusting the *LogFormat* directive associated with the *CustomLog* directive.

Another commonly used *LogFormat* for access log is "combined,,". See default *httpd.conf* file for details.

Instructor notes:**Purpose —****Details —** The common theme to the two examples within the chart are:

1. IP of the Client
2. Date/time stamp of the completion of the request
3. Request method
4. URI requested
5. Return code
6. Content length of the response

Additional information — Both IHS and WebSphere requests are recorded within the access log.

The default IHS logs directory is `<IHS Root>/logs`. Always refer to the `httpd.conf` file to determine the specific location and file name. It is possible to define multiple access logs and alternate paths.

Transition statement — Since the GET request was for a WebSphere application, you can research this further by reviewing the Plug-in log.

Identifying a working request: Plug-in log (1)

• GET request

[Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - TRACE:

┌──────────────────┬──────────────────┬──────────┐
 │ Date / time stamp │ Process / thread ID │ Message type │

- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - DETAIL: ws_common: websphereShouldHandleRequest: trying to match a route for: vhost='localhost'; uri='/PlantsByWebSphere'
- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - TRACE: mod_was_ap20_http: as_translate_name: WebSphere will handle: /PlantsByWebSphere
- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - DETAIL: ws_common: websphereFindTransport: Setting the transport(case 2): was61host01 on port 9080
- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - DETAIL: GET /PlantsByWebSphere HTTP/1.1
- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - TRACE: ws_common: websphereExecute: Wrote the request; reading the response
- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - DETAIL: lib_htresponse: htresponseRead: Reading the response: 529984
- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - DETAIL: HTTP/1.1 200 OK
- [Tue Jun 26 17:56:56 2007] 000002d8 000004a4 - DETAIL: ws_common: websphereEndRequest: Ending the request

© Copyright IBM Corporation 2007

Figure 15-8. Identifying a working request: Plug-in log (1)

WA5711.0

Notes:

The lines above are various log statements from a plug-in log with LogLevel set to trace. Plug-in trace is not enabled by default, in next few slides you will learn how to enable plug-in trace. These lines are not the complete trace of a request, but represent a few key entries to look for in order to validate that a request was received and processed by the plug-in.

The breakdown of each line is as follows:

- Shows that the request was received.
- Plug-in has successfully matched the request against its criteria outlined within the Plug-in-cfg.xml file.
- Plug-in chose Transport Hostname of App_Server_01 on Port 9080.
- Shows the type of method that will be submitted to the application server.
- The request has been submitted to the application server.
- The Plug-in is now reading the response from the application server.

7. Application Server posted a response code to the plug-in.
8. The request has ended.

To get this level of detail, you must enable tracing within the *Plug-in-cfg.xml* file:

```
<Log LogLevel="Trace" Name="C:\Program  
Files\IBM\WebSphere\Plugins/logs/webserver1/http_plugin.log"/>
```

The Name parameter provides the file name and location of the plug-in log.

Instructor notes:

Purpose —

Details — This is part of a plug-in log to show the necessary items to validate a working request. This does not represent the complete log for a request when Trace has been enabled.

Additional information — If there are problems locating the plug-in-cfg.xml file, review the httpd.conf file for the WebSpherePluginConfig directive (usually at the bottom) for the exact path to this file.

Transition statement — Next is Post.

Identifying a working request: Plug-in log (2)

• POST request



[Wed Jun 27 16:32:12 2007] 0000047c 00000874 - DETAIL:

[Wed Jun 27 16:32:12 2007] 0000047c 00000874 - DETAIL: ws_common: websphereShouldHandleRequest: trying to match a route for: vhost='localhost'; uri='/PlantsByWebSphere/servlet/AccountServlet'

[Wed Jun 27 16:32:12 2007] 0000047c 00000874 - TRACE: mod_was_ap20_http: as_translate_name: WebSphere will handle: /PlantsByWebSphere/servlet/AccountServlet

[Wed Jun 27 16:32:12 2007] 0000047c 00000874 - DETAIL: ws_common: websphereFindTransport: Setting the transport(case 2): was61host01 on port 9080

[Wed Jun 27 16:32:12 2007] 0000047c 00000874 - DETAIL: POST /PlantsByWebSphere/servlet/AccountServlet?action=login&updating=false HTTP/1.1

[Wed Jun 27 16:32:12 2007] 0000047c 00000874 - TRACE: ws_common: websphereExecute: Wrote the request; reading the response

[Wed Jun 27 16:32:12 2007] 0000047c 00000874 - DETAIL: lib_htresponse: htresponseRead: Reading the response: 52997c

[Wed Jun 27 16:32:30 2007] 0000047c 00000874 - DETAIL: HTTP/1.1 200 OK

[Wed Jun 27 16:32:30 2007] 0000047c 00000874 - DETAIL: ws_common: websphereEndRequest: Ending the request

© Copyright IBM Corporation 2007

Figure 15-9. Identifying a working request: Plug-in log (2)

WA5711.0

Notes:

These lines are almost identical to the previous GET slide. The only difference is the POST method was used.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — Any questions regarding this topic?

IHS and plug-in trace enabling

- Enabling IHS trace
 - Edit httpd.conf file
 - Change LogLevel from **warn** to **debug**
 - Restart IHS
 - Debug information will be written in error.log

- Enabling Plug-in trace
 - Edit plugin-cfg.xml file
 - Change LogLevel from **Error** to **Trace**
 - Restart Web server
 - Trace information will be written in http-plugin.log

© Copyright IBM Corporation 2007

Figure 15-10. IHS and plug-in trace enabling

WA5711.0

Notes:

1. It is not possible to change the log level for IHS dynamically. IHS instance must be restarted to pick up this change. Here is an extract from httpd.conf file for LogLevel directive:

```
#
# LogLevel: Control the number of messages logged to the error.log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn
```

2. Plug-in trace may be enabled dynamically. By default, RefreshInterval for plugin-cfg.xml file is 60 seconds. So, changes should get picked up dynamically. If changes are not picked up dynamically, you may have to restart the Web server. Here is an extract from plugin-cfg.xml for LogLevel:

```
<Log LogLevel="Error" Name="c:\IBM\WebSphere61\logs\http_plugin.log"/>
```

Available LogLevels are Error, Warning, Stats, Detail, Debug and Trace. Detail and Debug were introduced newly in V6.1. For plug-in problems that will take a long time to recreate, the user can try less verbose log level.

Instructor notes:

Purpose —

Details — There is no way to configure file rotation for plug-in. For Web sites with high traffic, trace may fill up file system. Also, these there might limits to file size. It is always recommended to make sure there is enough file system space before enabling the traces.

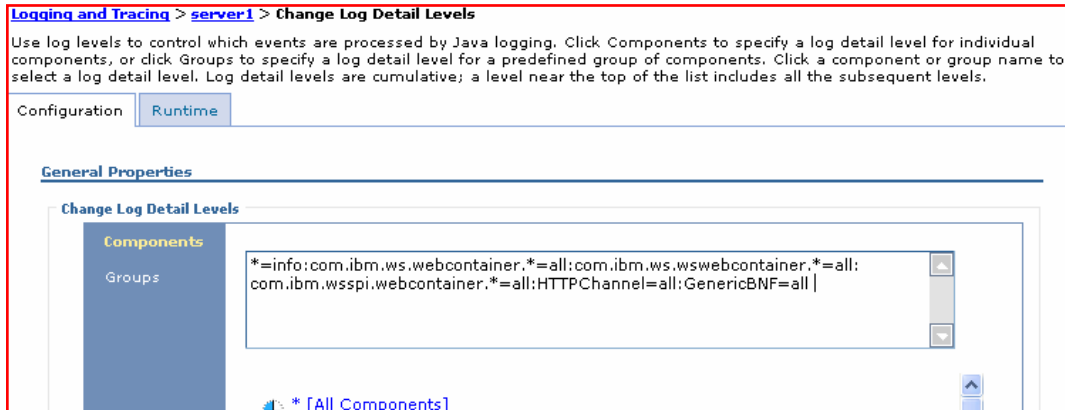
Additional information —

Transition statement — Any questions regarding this topic?

Enabling HTTP trace on the application server

- For WebSphere Application Server
 - Open Administrative console
 - Go to **Troubleshooting -> Logs and Trace -> Server_Name -> Change Log Level Details**
 - In **Configuration** Tab, enter below test in provided test box:


```
*=info:com.ibm.ws.webcontainer.*=all:com.ibm.ws.wswbcontainer.*=all:com.ibm.ws
spi.webcontainer.*=all:HTTPChannel=all:GenericBNF=all
```



- Save changes and restart the application server
- Trace information will be written into the trace.log of application server.

© Copyright IBM Corporation 2007

Figure 15-11. Enabling HTTP trace on the application server

WA5711.0

Notes:

1. You can enable trace for various components in application server. Above mentioned trace string enables traces for components that are of interest in handling the HTTP request. For detailed instructions to gather trace, you can refer to the URL:
<http://www-1.ibm.com/support/docview.wss?uid=swg21252138>
2. Trace can also be enabled dynamically. Above URL has instructions to collect trace dynamically as well.
3. File rotation can be configured in “Diagnostic Trace Service” page. You can specify file size and number of historical files to be kept. These should be specified according to the load on your system and time it takes to perform recreate.

Instructor notes:

Purpose —

Details —

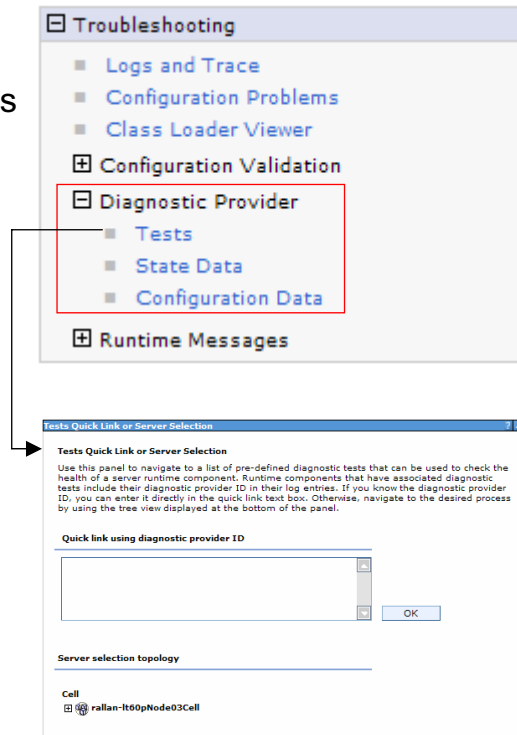
Additional information —

Transition statement — Any questions regarding this topic?

Web container diagnostic provider

- Web container diagnostic provider contains valuable information for debugging problems
 - Web container diagnostic providers are MBeans that provide configuration dump, state dump, and self-test routines

- Below listed are some important items that can be viewed:
 - Web Container settings
 - Virtual host aliases
 - Web Application specific information
 - Servlet mappings
 - Filter mappings
 - Listeners
 - Web application Cached targets information
 - Filter information
 - Listener information



© Copyright IBM Corporation 2007

Figure 15-12. Web container diagnostic provider

WA5711.0

Notes:

The IBM Education Assistant contains a module on the WebSphere Application Server V6.1 Diagnostic Providers that is highly informative and contains further resources regarding the diagnostic providers. This educational module is available as part of the WebSphere Application Server V6.1 Information Center using the URL http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v6/was/6.1/ProblemDetermination/WASv61_DiagnosticProviders/player.html.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting a failing request

After completing this topic, you should be able to:

- Identify preliminary questions to ask at the start of problem determination
- Determine what trace data is needed
- Review trace data to identify key failure points
- Make a rational choice to correct the problem

© Copyright IBM Corporation 2007

Figure 15-13. Troubleshooting a failing request

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — A checklist can be extremely helpful during a problem determination session. The following slides contain a set of questions that are helpful in tracking down the cause of the failed request.

Preliminary questions (1 of 2)

1. What is the error code in browser?
 - Possible ones are DNS Error, 4xx, 5xx or error message from application with no response code.
2. Are all components running?
 - Verify that WebServer and the WebSphere application are running.
3. DNS is working correctly?
 - Verify that the WebServer host name and the IP address of the box are set correctly within DNS.
4. What was the result of by-passing the Web server?
 - If the problem does not happen when by-passing the Web server, then there might be a Web server or plug-in problem.
5. Are there any firewall or front-end issues?
 - Know your topology and what components are between the client and the Web server. One of these could be keeping the request from going through.
6. What are the operating systems (OS) of all components involved?
 - Sometimes updates to components are published for your particular OS. Also, you may have an OS related patch that is required to correct a problem.

© Copyright IBM Corporation 2007

Figure 15-14. Preliminary questions (1 of 2)

WA5711.0

Notes:

Before you investigate a failing request, there are important questions you need to research first.

Question 1: Key question in further troubleshooting the problem. Response for this one help narrow down rest of the questions.

Question 2: This may seem like a dumb question, but there have been cases where the WebSphere Application Server is running, but the application is not.

Question 3: Make sure that the host name of the request does map to the IP of the Web server.

Question 4: Identify the likely location of the problem source. Is it in the Web server or is it in the application server?

Question 5: Firewalls have to be configured to allow communications to the intended Web server or application server. Verify that they are open to the host name/IP address and port number of the request.

Question 6: When researching problematic requests, sometimes it becomes apparent that the proper maintenance updates for the operating system have not been applied.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement — More questions are on the next slide.

Preliminary questions (2 of 2)

7. What are the versions and maintenance levels of the involved products?
 - Important for researching possible updates to the Web server, plug-in and application server.
8. What did you type in?
 - Need to know the exact request that was typed into the client browser.
9. What did you get back?
 - Get a print screen of the results. Problems can result with the wrong response or no response.
10. What was the date and time of the request?
 - This aids in locating the problematic request within the logs.
11. Who was supposed to respond to the request?
 - Web server or application server

© Copyright IBM Corporation 2007

Figure 15-15. Preliminary questions (2 of 2)

WA5711.0

Notes:

Question 7: Like question 6, problematic requests may be documented and resolved in later releases of your Web server, plug-in, or application server updates.

Question 8: Knowing the exact request typed into the client browser is critical to diagnosing the problem. This also determines if the request was SSL or not.

Question 9: Need to have verification of the results. You will use this to determine what kind of return code may have been issued.

Question 10: Key item for reviewing long logs

Question 11: This question is key to identifying what logs will be necessary to troubleshoot the problem. Knowing your application architecture could expedite where to search first.

Instructor notes:

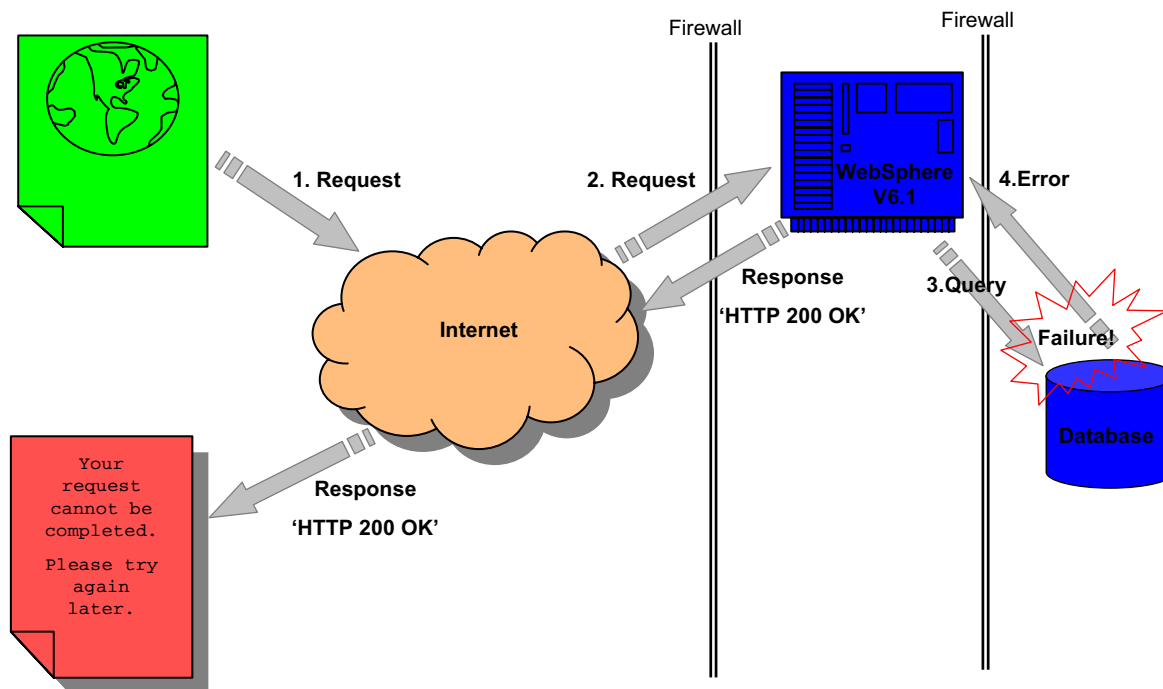
Purpose —

Details —

Additional information —

Transition statement —

A word about HTTP response codes



© Copyright IBM Corporation 2007

Figure 15-16. A word about HTTP response codes

WA5711.0

Notes:

The HTTP response codes might not be helpful in tracking down the problem because all response messages could be '200 OK'. This is because it is common for well designed enterprise Web sites to handle errors at the Web server or application server and return a user friendly Web page to the customer, explaining the nature of the problem. As a result, Web site monitoring tools will typically provide configuration parameters that take strings as input, and trigger an event based on finding that string in the Web page. For example, if a database server is unavailable and the transaction fails as a result, the monitoring tool will look through the returned Web page for something like "Unable to complete your request" or some error message meant to inform the user that a problem has occurred.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web server generating a 404 response

- IBM HTTP Server example for the following request:
 - `http://localhost/index99.html`

```
Response Header
HTTP/1.0 404 Not Found
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354
```

- Locate the request within the `<IHS Root>/logs/access.log`
 - `127.0.0.1 - - [26/Jun/2007:17:49:59 -0400] "GET /index99.html HTTP/1.1" 404 280`
- Locate the error within the `<IHS Root>/logs/error.log`
 - `[Tue Jun 26 17:49:59 2007] [error] [client 127.0.0.1] File does not exist: C:/Program Files/IBM/HTTPServer/htdocs/en_US/index99.html`

© Copyright IBM Corporation 2007

Figure 15-17. Web server generating a 404 response

WA5711.0

Notes:

The request for `mypage.html` produced a 404 response. The access log clearly shows that it posted the 404 response. The error log shows that it attempted to serve the request from its document root (`C:/Program Files/IBM HTTP Server/htdocs/en_US`) and could not find the requested file.

It is recommended that you always start reviewing the Web server logs first when debugging non-working requests. Requests served only by the Web server are classified as static requests and usually consist of HTML pages and image files. Requests served by WebSphere are classified as dynamic requests, but may also include static content. For the purpose of this unit, content referenced as static is served by the Web server while content referenced as dynamic is served by the application server.

The following are key things to know ahead of time:

1. What file was requested (for example, `mypage.html`)?
2. Date/time the problem occurred
3. The IP address of the client browser

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web server generating a 404 response: Solution

- Verify the documentRoot path in the httpd.conf configuration file (<IHS_root>/conf/httpd.conf) in the IBM HTTP Server is correct
 - For example;
 - DocumentRoot “C:/Program Files/IBM HTTP Server/htdocs/en_US”
- Confirm that the file mypage.html exists under that directory.
 - The path to mypage.html will be derived based on the documentRoot value and the relative URI and any URL rewriting that occurs as a result of the request
 - In a localized environment, the path to the resource could be qualified by the locale
 - A resource sitting in the documentRoot will not be located if the URL is rewritten to include the locale
 - For example, a request for somedir/mypage.html would need to be in the directory somedir/en_US/mypage.html after the URL localization occurs for a user agent request the US_en version of the page

© Copyright IBM Corporation 2007

Figure 15-18. Web server generating a 404 response: Solution

WA5711.0

Notes:

If the static content you are requesting does not exist under the *DocumentRoot*, then a “404 Not Found” error will occur. You must make sure your content exists under the *DocumentRoot* to avoid these types of errors. You may also change the *DocumentRoot* path if needed.

Possible causes of a 404 error include:

- File is missing from specified path
- Path is incorrect
- Spelling or casing of the request is wrong

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Plug-in fails to route a request to application server

- “404 Not Found” generated by the IBM HTTP Server when the request should have been sent to WebSphere:
 - `http://localhost/PlantsByWebSphere`
- In the IBM HTTP Server, locate the request within the `<IHS Root>/logs/access.log`:
 - `127.0.0.1 - - [26/Jun/2007:17:59:01 -0400] "GET /PlantsByWebSphere HTTP/1.1" 404 305`
- Search for “File does not exist” errors in the `<IHS Root>/logs/error.log`:
 - `[Tue Jun 26 17:59:01 2007] [error] [client 127.0.0.1] File does not exist: C:/Program Files/IBM HTTP Server/htdocs/en_US/PlantsByWebSphere`

© Copyright IBM Corporation 2007

Figure 15-19. Plug-in fails to route a request to application server

WA5711.0

Notes:

This example shows that the Web server attempted to serve a WebSphere request from its document root.

A “File does not exist” errors for a particular URI request (for example, `/PlantsByWebSphere`) recorded in the `error.log` is a clear indication that the HTTP plug-in failed to send the request to WebSphere. Instead, the HTTP plug-in gave the request back to the Web server to process. The Web server will then try to serve the request from its `DocumentRoot` where it obviously does not exist. This is usually caused by an incorrect configuration of the `plug-in-cfg.xml` file. See the next slide.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Plug-in causing a 404 response: HTTP Plug-in

- Locate the request in the http_plugin.log to determine why the HTTP plug-in failed to send the request to WebSphere:
 - [Tue Jun 26 17:59:01 2007] 000002d8 000004a8 - DETAIL:
ws_common: **websphereShouldHandleRequest: trying to match a route for: vhost='localhost'; uri='/PlantsByWebSphere'**
 - [Tue Jun 26 17:59:01 2007] 000002d8 000004a8 - TRACE:
ws_common: websphereVhostMatch: Comparing '*:80' to 'localhost:80' in VhostGroup: default_host
 - [Tue Jun 26 17:59:01 2007] 000002d8 000004a8 - DEBUG:
ws_common: websphereVhostMatch: Found a match '*:80' to 'localhost:80' in VhostGroup: default_host with score 1, exact match 0
 - [Tue Jun 26 17:59:01 2007] 000002d8 000004a8 - TRACE:
ws_common: **websphereUriMatch: Failed to match: /PlantsByWebSphere**
 - [Tue Jun 26 17:59:01 2007] 000002d8 000004a8 - DETAIL:
ws_common: websphereShouldHandleRequest: **No route found**

© Copyright IBM Corporation 2007

Figure 15-20. Plug-in causing a 404 response: HTTP Plug-in

WA5711.0

Notes:

The above plug-in trace clearly shows that the plug-in was not able to find a valid URI match for /PlantsByWebSphere under the UriGroup: Web_Application_URIs, and therefore was unable to determine the route. A “No route found” message is then recorded, and the plug-in gives the request back to the Web server. At this point, the Web server tries to serve the request from its document root as seen in the previous slide. The result is a 404 error generated by the Web server.

An application server may return 404 error for various reasons. Here are few:

- Requested (or dependent) resource not found in Web application
- Servlet handling the request sets 404 as response status code
- Requested application was stopped
- Application mapped to a wrong virtual host or invalid alias in mapped virtual host.
- Plug-in-cfg.xml contains wrong information and routes request to wrong server.

- Use WebContainerDP to view configuration data of Web container. This should contain all information about virtual host mappings.
- For all cases where application server returning 404, you will see information similar to below in plug-in trace:

```
[Tue Jun 26 17:57:48 2007] 000002d8 000004a8 - DEBUG: lib_htrequest:
htrequestWrite: Writing the request:
[Tue Jun 26 17:57:48 2007] 000002d8 000004a8 - DETAIL:      GET
/PlantsByWebSphere/GetPlant.jsp HTTP/1.1
[Tue Jun 26 17:57:48 2007] 000002d8 000004a8 - DETAIL:      Accept: image/gif,
image/x-xbitmap, image/jpeg, image/pjpeg, */*
... ..
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      HTTP/1.1 404 Not
Found
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      Content-Type:
text/html; charset=ISO-8859-1
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      $WSEP:
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      Content-Language:
en-US
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      Transfer-Encoding:
chunked
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      Connection: Close
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      Date: Tue, 26 Jun
2007 21:57:51 GMT
[Tue Jun 26 17:57:51 2007] 000002d8 000004a8 - DETAIL:      Server: WebSphere
Application Server/6.1
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Application server generating a 404 response

- A requested (or dependent) resource not found in Web application
- The symptoms include
 - 404 return code in access.log
 - Plug-in trace shows application server sending 404 response
 - SystemOut.log may not contain any information
 - Application server trace shows similar information as below:

```
[6/19/07 21:50:38:274 EDT] 0000001d WebApp      3   Exception type=java.io.FileNotFoundException
[6/19/07 21:50:38:274 EDT] 0000001d WebApp      3   Exception message=SRVE0190E: File not found:
/dummyTest
```

- Possible solutions include
 - Verify that the URL is valid, the context root of Web application, the file exist in Web application, and the resource defined in web.xml
- Important:
 - Verify that the correct version of the plug-in.xml file is being referenced, and not an older version

© Copyright IBM Corporation 2007

Figure 15-21. Application server generating a 404 response

WA5711.0

Notes:

Web container diagnostic providers information include:

1. Static content and HTML files, such as HTML, are not served.
 - Review configDump of WebContainerDP and verify that fileServingEnabled is set to true for the application in question.
 - Example: `startup-vhosts-admin_host-webapps-filetransfer#filetransfer.war-fileServingEnabled = true`
2. Welcome files are not served.
 - For Welcome files to be served using Web server plug-in, fileServing feature should be enabled. The same information in the above case should help.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Plug-in generating a 500 response: Solution

- A Web or application server 500 level error can be solved by inspecting:

```
Response Header
HTTP/1.0 500 Internal Server Error
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354
```

- The network for any performance or DNS issues.
- The operating system TCP/IP layer
- The WebSphere Application Server to ensure the server is not hung or busy
- The WebSphere Application Server to ensure the server was properly started and open for e-business
- The plugin-cfg.xml ConnectTimeout and ServerIOTimeout settings to ensure they are not set too low
- Any firewalls or switch devices which sit in between the plug-in and WebSphere Application Server to ensure they are working properly and will allow traffic to and from the back-end transport ports (for example, 9080, 9443)

© Copyright IBM Corporation 2007

Figure 15-22. Plug-in generating a 500 response: Solution

WA5711.0

Notes:

This error is typically seen when the HTTP plug-in is not able to communicate with a back-end WebSphere Application Server V6.1. It can also be returned by a WebSphere application server when an error is encountered while processing a request (for example, servlet or JSP page).

The “Internal Server Error” message can be produced in one of three ways:

1. In rare cases when using CGI scripts within the Web server, a 500 can be produced if it has a problem with the script.
2. The plug-in is unable to make a connection to any cluster member.
3. The application server encountered an error when processing the request.

In the situations described above, a packet trace is very useful to determine where breakdowns occur. Use the URL below to get the details on how to set up a packet trace for your operating system.

<http://www-1.ibm.com/support/docview.wss?uid=swg21175744>

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

HTTP Plug-in 500 level generated

- Locate GET request in *http_plugin.log*
 - [Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL: GET
/PlantsByWebSphere HTTP/1.1
- Locate the response in the plug-in log

```
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:
lib_htresponse: htresponseRead: Reading the response: 837c74
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:
HTTP/1.1 500 Internal Server Error
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:      Date:
Sat, 17 Mar 2007 22:01:51 GMT
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:      Server:
WebSphere Application Server/6.1
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:
Content-Type: text/xml; charset=UTF-8
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:
Content-Language: en-US
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:
Transfer-Encoding: chunked
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:
Connection: Close
```

© Copyright IBM Corporation 2007

Figure 15-23. HTTP Plug-in 500 level generated

WA5711.0

Notes:

This slide shows an example of a 500 response within the Plug-in log sent by the WebSphere application server.

As indicated earlier in 404 Scenario 1, when reviewing the plug-in log, the key strings to search on are the URI */PlantsByWebSphere* and the date/time of the failed request. The above trace entries show that the 500 Internal Server Error response came from the WebSphere application Server. The 500 response indicates that WebSphere encountered an internal processing error when attempting to serve the *PlantsByWebSphere* page and returned the 500. This 500 Internal Server Error is returned to the client browser.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere Application Server 500 response

- Search the WebSphere Application Server V6.1 *trace.log* for 500 response codes with the same date/time stamp seen in the plug-in log.
- Then, review the entire thread (e.g. 3605eb2e) to learn more about how the request was processed

```
[4/19/07 13:00:00:670 EST] 3605eb2e WebApp      d Exception
errorCode=500
[4/19/07 13:00:00:670 EST] 3605eb2e WebApp      d Exception
type=java.lang.IllegalStateException
[4/19/07 13:00:00:670 EST] 3605eb2e WebApp      d Exception
message=OutputStream already obtained
[4/19/07 13:00:00:670 EST] 3605eb2e WebApp      d Found
exception type=java.lang.IllegalStateException with location=null
```

© Copyright IBM Corporation 2007

Figure 15-24. WebSphere Application Server 500 response

WA5711.0

Notes:

This slide shows an example of the WebSphere application server trace showing the 500 error code.

The above trace entry shows that a `java.lang.IllegalStateException` was encountered when processing the request, and, as a result, a 500 error code was generated. Unfortunately, some exceptions that occur when processing a request may generate a 500 response, but not record an `errorCode=500` entry in the trace log. Regardless, locating the exception message for a particular request is key to moving forward in helping to determine the cause of a 500 response.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Solving a 500 response

- “500 Internal Server Error” responses are usually more difficult to resolve because their reasons are vast and sometimes complex.
- If the WebSphere Application Server returns a 500 response, this is an indication that something is seriously wrong within the application code or configuration.
- For more information on debugging 500 “Internal Server Error” messages visit the WebSphere Application Server support page and search on keywords “status code 500”
 - <http://www.ibm.com/software/webservers/appserv/was/support/>
 - There are numerous technotes which discuss solutions to known issues and defects related to the 500 response code.

© Copyright IBM Corporation 2007

Figure 15-25. Solving a 500 response

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Describe the Web Request flow
- List various ways of serving a request
- Enable IHS and Plug-in trace
- Enable application server trace
- Use the Web Container Diagnostic Provider
- Troubleshoot a failing request

© Copyright IBM Corporation 2007

Figure 15-26. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 10: Troubleshooting request flow and Web container problems

© Copyright IBM Corporation 2007

Figure 15-27. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 16. How to troubleshoot Web services

Estimated time

00:45

What this unit is about

This module explains how to detect, determine the root cause of, and resolve problems that exist with Web services.

What you should be able to do

After completing this unit, you should be able to:

- Describe the basics of Web services
- Detect problems with Web services during the development, deployment, and runtime phases
- Determine the root cause of a Web service problem
- Resolve the problem

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

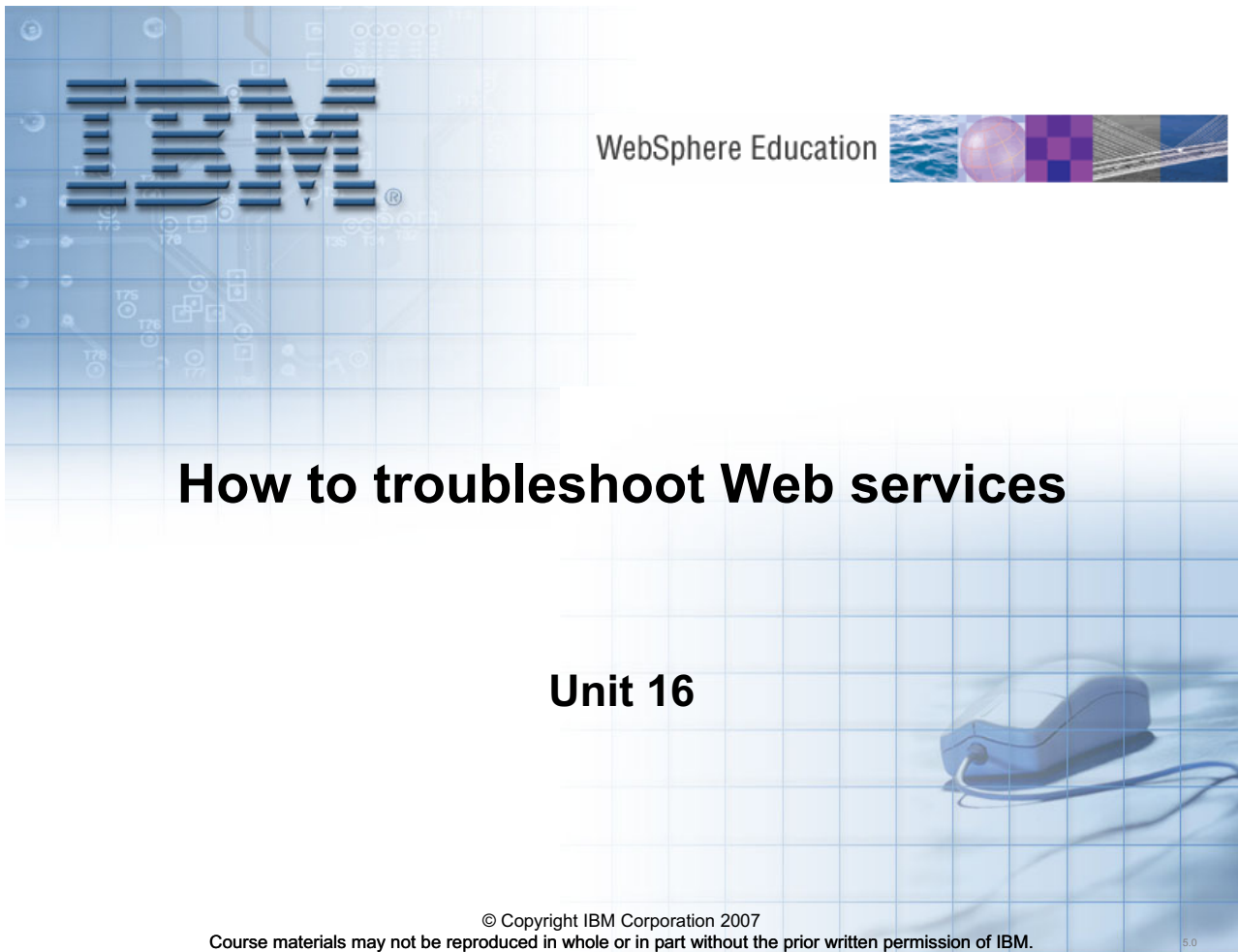


Figure 16-1. How to troubleshoot Web services

WA5711.0

Notes:

Instructor notes:

Purpose —

Details — Estimated time

00:45

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Understand Web services from a high level
 - SOA and Web services
 - The components and activities that make up Web services
 - Tools available for generating Web services
- Identify the common problems associated with developing and deploying Web services
 - Development problems
 - Deployment problems
 - Runtime problems
- Determine the root cause of Web services problems and resolve them
 - Code generation and compiler problems
 - Binding problems
 - Protocol problems

© Copyright IBM Corporation 2007

Figure 16-2. Unit objectives

WA5711.0

Notes:

The focus of this module is WebSphere Application Server v6.1. However, the utilities and development environments play a large part in preparing the Web services for deployment. Therefore, the development phase will be discussed in order to provide a more complete understanding of the Web service process.

Much of the information used in this module is drawn from various published references available through the IBM Web site. A reference to the documentation and links to the information is included at the end of the module.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

SOA and Web services

- Web services are a *realization* of SOA
 - A Web service is the implementation of a portion of the SOA business process
- IBM has defined 5 entry points into an SOA solution
 - People – automating the human tasks
 - Process – the way that the company does their business
 - Information – the knowledge and data that a company acquires as a result of its business
 - Connectivity – the integration of the people, processes, and information
 - Reuse – save resources by leveraging what is already there
- Entry points were defined to help customers understand SOA
 - These entry points identify the importance of the relationship between the business and technical stakeholders

© Copyright IBM Corporation 2007

Figure 16-3. SOA and Web services

WA5711.0

Notes:

IBM description of SOA

Service-oriented architecture (SOA) is a business-centric IT architectural approach that supports integrating your business as linked, repeatable business tasks, or services. SOA helps users build composite applications, which are applications that draw upon functionality from multiple sources within and beyond the enterprise to support horizontal business processes'

IBM has defined 5 entry points through real customer experience. The goal of SOA is to facilitate the automation of business processes to serve stake holders from both business and IT backgrounds: Business stakeholders bring domain expertise and technical stakeholders supply the skills to automate those processes.

SOA is managed through a set of IT rules known as *SOA Governance* that are focused on the *decision rights* involved with development, deployment, and management of Web services. SOA governance is concerned with best practices and IT processes.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web services component overview

- Universal Description, Discovery and Integration (UDDI)
 - A registry used to publish Web Services Description Language WSDL representations of Web services
 - Can be hosted in any combination of application server, database, DNS server, LDAP server, and so on.
- Client
 - Thick, thin, or middleware client
 - Implemented in J2EE or .NET
 - Hosted in a Web browser, in an application server, or as a Java client
- Web service
 - A new or legacy component that automates some or all of a business process and is described using WSDL

© Copyright IBM Corporation 2007

Figure 16-4. Web services component overview

WA5711.0

Notes:

Web services is a technology neutral component model. The implementation language and technologies should not be a limiting factor in realizing Web services and a service-oriented architecture. Two common technologies used to implement Web services are J2EE or Microsoft .NET frameworks. As a result, the set of Web service tools available are usually based in one of these paradigms. IBM fully supports the J2EE/Java-based approach with its tools but places no restrictions on clients or providers implemented in other technologies, as long as they are compliant with the Web services standards.

The boundary between a Web service and a client can be subtle. For example, a Web service that returns a translation from one language to another might proxy client requests to a completely different Web service or a legacy application on a mainframe. There are no restrictions on how a Web service is implemented, only on how it is described using WSDL.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web services component interaction overview

- Publish
 - Eligible Web services are published to a UDDI server
 - WSDL is used to describe the service
 - The WSDL represents the server side implementation of the Web service and contains the service endpoint URL so that clients can bind to the service
- Discover
 - A UDDI server can be queried for Web services
 - WSDL contains the attributes of the Web service
- Bind
 - The client uses the WSDL binding information to connect to the Web service

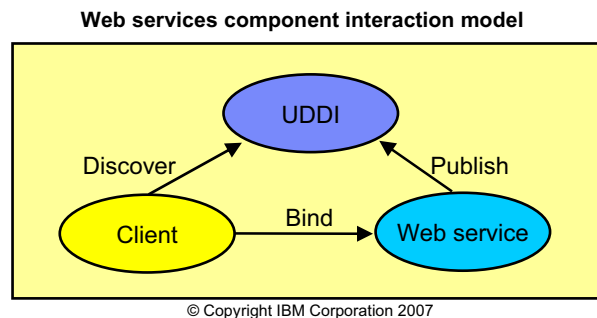


Figure 16-5. Web services component interaction overview

WA5711.0

Notes:

The *Publish/Discover/Bind* component diagram should be familiar to anyone that has followed the evolution of Web services. Web service components and activities are centered on the Web Service Description Language (WSDL) file. The WSDL file is the contract that describes what the Web service does, where it is located, and how the communication takes place. The diagram places no restrictions on where the components are designed or how they are implemented. The client, Web service, and UDDI registry could all be hosted on WebSphere Application Server v6.1 and is fully supported by the application server.

Instructor notes:**Purpose —****Details —****Additional information —**

Transition statement — The *Publish/Discover/Bind* diagram is the formal representation of Web services. But what do the actual pieces look like that make Web services work together?

Web Services Description Language (WSDL) elements

Example WSDL File

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://simplifiedate.ibm.com" xmlns:impl="http://simplifiedate.ibm.com" x
<wsdl:types>
<schema targetNamespace="http://simplifiedate.ibm.com" xmlns="http://www.w3.org/2001/XMLSchema" xmlns:t
<element name="getCurrentDateResponse">
<complexType>
<sequence>
<element name="getCurrentDateReturn" nillable="true" type="xsd:dateTime"/>
</sequence>
</complexType>
</element>
<element name="getCurrentDate">
<complexType>
<sequence/>
</complexType>
</element>
</schema>
</wsdl:types>
<wsdl:message name="getCurrentDateResponse">
<wsdl:part element="impl:getCurrentDateResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="getCurrentDateRequest">
<wsdl:part element="impl:getCurrentDate" name="parameters"/>
</wsdl:message>
<wsdl:portType name="SimpleDate">
<wsdl:operation name="getCurrentDate">
<wsdl:input message="impl:getCurrentDateRequest" name="getCurrentDateRequest"/>
<wsdl:output message="impl:getCurrentDateResponse" name="getCurrentDateResponse"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="SimpleDateSoapBinding" type="impl:SimpleDate">
<wsaw:UsingAddressing wsdl:required="false" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"/>
<wsdl:soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="getCurrentDate">
<wsdl:soap:operation soapAction="getCurrentDate"/>
<wsdl:input name="getCurrentDateRequest">
<wsdl:soap:body use="literal"/>
</wsdl:input>
<wsdl:output name="getCurrentDateResponse">
<wsdl:soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="SimpleDateService">
<wsdl:port binding="impl:SimpleDateSoapBinding" name="SimpleDate">
<wsdl:soap:address location="http://localhost:9080/SimpleDateService/services/SimpleDate"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

© Copyright IBM Corporation 2007

Figure 16-6. Web Services Description Language (WSDL) elements

WA5711.0

Notes:

The WSDL document is the *glue* that holds the Web service components together and includes elements such as:

service element

The relationship between the Web service and its location

- *portType* element

The set of operations and their messages

- *operation* element

A specific method

- *message* element

The input and output parameters to the Web service

- *types* element

The data types used expressed using XML schema

- *binding* element

The communications protocol for calling the operation

The *Web Services Description Language* (WSDL) document is based on a XML schema and contains elements and attributes that describe the Web service interface, binding, operations, and messages supported by the Web service. Typically, the WSDL document is hosted on the Universal Description, Discovery and Integration (UDDI) server and clients can be generated from the WSDL file.

Think of a *portType* as mapping (loosely) to a Java class with the *operation* and *message* elements mapping to methods and parameters, respectively. The *binding* element describes the transport and communication parameters used in communicating with the Web service.

Service element example

```
<wsdl:service name="DateFormatterService">
  <wsdl:port binding="intf:DateFormatterSoapBinding" name="DateFormatter">
    <wsdlsoap:address
      location="http://localhost:9084/DateFormatterWeb/services/DateFormatt
        er" />
    </wsdl:port>
  </wsdl:service>
```

portType element example

```
<wsdl:portType name="DateFormatter">
  <wsdl:operation name="getFormattedDate">
    <wsdl:input message="intf:getFormattedDateRequest"
      name="getFormattedDateRequest" />
    <wsdl:output message="intf:getFormattedDateResponse"
      name="getFormattedDateResponse" />
  </wsdl:operation>
</wsdl:portType>
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere Application Server v6.1 hosting environment for Web services

- WebSphere Application Server v6.1 can host Web service providers, Web service clients, and UDDI registries
 - WebSphere Application Server v6.1 includes a UDDI registry component for hosting published Web service descriptions as part of the installation
- WebSphere Application Server v6.1 supports Web services based on Enterprise JavaBeans (EJBs) or standard JavaBeans
 - A EJB based Web service is based on a Remote Method Invocation (RMI) binding
 - A standard JavaBean Web service uses a servlet based mechanism and HTTP binding
- A Web service client can be hosted as an EJB or Web application capable of interacting with other Web services

© Copyright IBM Corporation 2007

Figure 16-7. WebSphere Application Server v6.1 hosting environment for Web services

WA5711.0

Notes:

Rational Application Developer, Application Server Toolkit, and WebSphere Application Server v6.1 include various Web service utilities for developing, configuring, and running Web services. Additionally, various command line utilities are available for working with Web services, using WSDL files, JavaBeans, or EJBs as input. The batch files are located in the `<install_root>/bin` directory of the various development and middleware offerings.

This module looks at Web services hosted in WebSphere Application Server v6.1. However, it should be kept in mind that Web services are based on open standards and many middleware offerings include support for Web services, such as databases and messaging software.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere code generation tools

- IBM offers a variety of Web service tooling and wizards
 - *Rational Application Developer* (RAD) can be used to generate Web services from Java Beans, EJBs, or WSDL files
 - Multiple command line scripts are available to cover client, server, and publishing scenarios
- Web service utilities ship with various IBM development, deployment, and hosting products. Some of the more common scripts are:
 - WSDL2Java
 - A command line script that generates Java client bindings from a WSDL file
 - Java2WSDL
 - A command line script that generates a WSDL file from a Java class
 - WSDL2Client
 - A command line script that generates a Web services client from one or more WSDL files

© Copyright IBM Corporation 2007

Figure 16-8. WebSphere code generation tools

WA5711.0

Notes:

In addition to the WSDL2Java and Java2WSDL tools the IBM Eclipse-based development environments include tools such as:

WSDL2WebService

- Generates a Web service from a WSDL file using the WebSphere runtime environment
- Bean2WebService
 - A command line tool that generates a Web service from a standard JavaBean
- EJB2WebService
 - A command line tool that generates a Web service from a stateless enterprise session bean

By default, the Web services created are based on J2EE 1.4. However, each of the command line tools also support Web service generation based on J2EE 1.3. These scripts

end in the token *13*. For example, *WSDL2Java13.bat* will generate a JavaBean from a WSDL file that is J2EE 1.3 compliant.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Artifacts created during code generation

- Code generation utilities and wizards produce various Java classes and XML files
 - For example, starting with a simple Java Bean and generating a Web services results in the artifacts
 - An interface class
 - A WSDL file
 - A *webservices.xml* file
 - A Web service mapping file (xml)
 - IBM Web service extensions

JavaBean

```

package com.sw571.example;

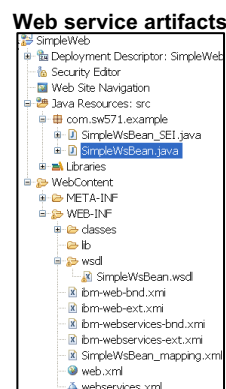
import java.util.Date;

public class SimpleWsBean {
    private String formattedDate;

    /**
     * no args constructor for Java Bean compliance
     */
    public SimpleWsBean() {
    }

    /**
     * get the date string
     *
     * @return String
     */
    public String getFormattedDate() {
        formattedDate = new Date().toString();
        return formattedDate;
    }
}

```



© Copyright IBM Corporation 2007

Figure 16-9. Artifacts created during code generation

WA5711.0

Notes:

Rational Application Developer Version 7.0.0 was used to generate the components of this simple Web service, pictured in Figure 2. The example JavaBean used to demonstrate the code generation is as simple as possible in order to demonstrate the various artifacts necessary for successful deployment of a Web service. Here are some of the details associated with the artifacts created during the code generation:

1. An interface class – The interface generated, `com.sw571.example.SimpleWsBean_SEI.java`, includes the method signature that conforms to the JavaBean
2. A WSDL file – The WSDL file generated `SimpleWeb/WebContent/WEB-INF/wsdl/SimpleWsBean.wsdl`, includes the message, binding, portType, operation, service, and port elements that describe the Web service
3. A *webservices.xml* – This file describes the Web service
4. A Web service mapping file – This file, `SimpleWsBean_mapping.xml`, is a mapping between the Web application and the WSDL file

5. IBM Web service extensions – These files, `ibm-webservices-bnd.xmi` and `ibm-webservices-ext.xml`, are IBM specific binding files

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web services security

- Message level integrity, confidentiality, and authentication are supported by extensions to SOAP
 - Integrity
 - Achieved using XML Digital Signature technology
 - Confidentiality
 - Achieved using XML Encryption
 - Authentication
 - Achieved using credentials such as a username and password

- The SOAP standard provides support for encryption and digital signing of SOAP messages
 - Web services SOAP messages can be encrypted, digitally signed, or both

- A common security scenario is to send clear text SOAP messages across an SSL encrypted stream coupled with HTTP basic authentication

© Copyright IBM Corporation 2007

Figure 16-10. Web services security

WA5711.0

Notes:

Here are some examples of SOAP requests using various forms of security:

1. Basic authentication

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-w
ssecurity-secext1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>
          myusername
        </wsse:Username>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
</soapenv:Envelope>
```

```

    <wsse:Password
      Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">
      *****
    </wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>
</soapenv:Header>
<soapenv:Body>
  <p947:getTheTime xmlns:p947="http://time.ws.swat.wst.ibm.com"/>
</soapenv:Body>
</soapenv:Envelope>

```

This SOAP request contains the username and password in the header of the SOAP envelope. An encrypted stream is crucial to protect the user credentials. Such a request would be used in conjunction with SSL/TLS1.0.

2. SOAP request using XML encryption

```

<?xml version='1.0'?>
<soapenv:Envelope xmlns:soapenc="http://schemas..."
  xmlns:soapenv="http://schemas..." xmlns:xsd="http://www.w3.org..."
  xmlns:xsi="http://www.w3.org...">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
      xmlns:wsse="http://docs.oasis-open...xsd">
      <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
        <EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
              ValueType="http://docs...-profile-1.0#X509SubjectKeyIdentifier">
              /62wXObED7z6c1yX7QkvN1thQdY=
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <CipherData>
          <CipherValue>
            ...lMSxrHeMQUN81Ubv2...R24KI...q+jUGV+jLoyKbK0i0Md7DeI...4KfduGJhcXL...gRDB
            lY83P...GmbM...3RoHrU/+2ng=
          </CipherValue>
        </CipherData>
        <ReferenceList>
          <DataReference URI="#wssecurity_encryption_id_855531804191795449"/>

```

```

        </ReferenceList>
        </EncryptedKey>
    </wsse:Security>
</soapenv:Header>
<soapenv:Body>
    <EncryptedData Id="wssecurity_encryption_id_..."
    Type="http://www.w3.org/2001...."
    xmlns="http://www.w3.org/.../xmlenc#">
    <EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>
    <CipherData>
        <CipherValue>

r49K72Q6yc+afp5t9H0bvHYNsUso...2bab04tLtuc5nUo+dbrCRpM5Zo...GOVQthNIUeGjHf
        </CipherValue>
    </CipherData>
    </EncryptedData>
</soapenv:Body>
</soapenv:Envelope>

```



Note

The value of the *CipherValue* elements in the header and the body have been shortened to fit the page.

These SOAP header contains the cypher text used by the sender to decrypt the message using the receivers public key. The SOAP body contains the cypher data containing the message. In this way, the SOAP message can be sent over clear text, securely.

In addition, SOAP supports XML digital signature technology so the receiver can be assured that the message came from a trusted sender.

Instructor notes:

Purpose —

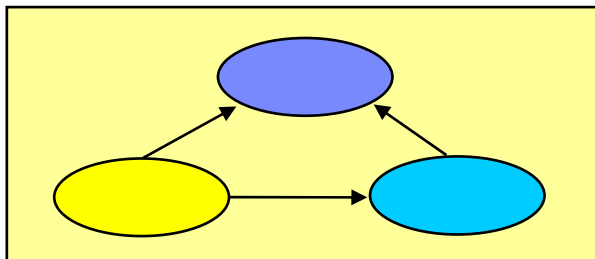
Details —

Additional information —

Transition statement —

Checkpoint

1. Name the 5 SOA entry points defined by IBM
2. Name the Web service components and their interactions



3. True or False: A Web service can be based on an Enterprise JavaBean.
4. True or False: SOAP messages provide security mechanisms for digital signing but not XML encryption.

© Copyright IBM Corporation 2007

Figure 16-11. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1.
 - a.
 - b.
 - c.
 - d.
 - e.
2.
 - a.
 - b.
 - c.
3. True or false? Explain.

4. True or false? Explain.

Instructor notes:

Purpose —

Details — Answers:

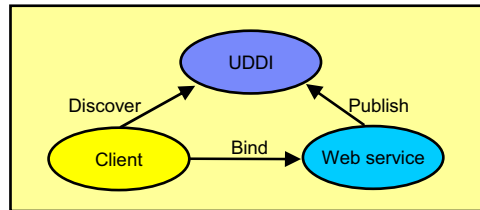
1. People, process, connectivity, information, re-use.
2. The components are Web service, client, and UDDI. Their interactions are publish, discover, and bind.
3. True. Web services can be based on Enterprise Java Beans and standard JavaBeans.
4. False. SOAP messages can be digitally signed and they can be encoded.

Additional information —

Transition statement —

Checkpoint solutions

1. Name the 5 SOA entry points defined by IBM
 - **People, process, connectivity, information, re-use.**
2. Name the Web service components and their interactions
 - **The components are Web service, client, and UDDI. Their interactions are publish, discover, and bind.**



3. True or False: A Web service can be based on an Enterprise JavaBean.
 - **True. Web services can be based on Enterprise Java Beans and standard JavaBeans.**
4. True or False: SOAP messages provide security mechanisms for digital signing but not XML encryption
 - **False. SOAP messages can be digitally signed and they can be encoded.**

© Copyright IBM Corporation 2007

Figure 16-12. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic objectives

After completing this topic you should be able to:

- Describe some of the common problems that can arise with Web services such as:
 - Web service code generation problems using the workbench wizards and command line tools
 - Binding problems associated with WSDL creation when using workbench wizards and the command line tools
 - Client/server problems between .NET and J2EE

© Copyright IBM Corporation 2007

Figure 16-13. Topic objectives

WA5711.0

Notes:

A large class of problems, for any enterprise technologies, are related to security problems such as authentication failures, certificate expiration, role mapping, and many others. This class contains a module dedicated to solving security issues. In many cases the security problems encountered with any Web application can also occur in a Web service. During run time the following classes of problems could occur:

Authentication or authorization failure

- Secure Web service fails to start
- Interoperation problems between WebSphere Application Server v6.1 and previous versions of WebSphere Application Server

Such problems might have been introduced during the development or deployment of the Web service, not becoming evident until the loosely bound parameters become resolved during run time.

Web server security troubleshooting tips

As with any security problem, it is important to verify that the basic administrative security of the environment is functioning correctly before delving into the security rules associated with the application security policies and parameters. For example, telnet to the host machine and confirm that you can authenticate. Additionally, try logging into the administration console or generating a simple, secure request from a test environment to confirm that the application server is accessible through the application servers login module, with the credentials that the Web service employs.

When using a trace string to assist with debugging Web service security issues, the IBM Information Center suggests the following configuration:

```
com.ibm.xml.soapsec.*=all=enabled:com.ibm.ws.webservices.*=all=enabled:
com.ibm.wsspi.wssecurity.*=all=enabled:com.ibm.ws.security.*=all=enabled:
SASRas=all=enabled
```

1. "WSE567: The incoming Username token must contain both a nonce and a creation time for the replay detection feature" Microsoft .NET error displays

This error is actually generated by a .NET server responding to a J2EE client. Configuring both a nonce and a time stamp is optional for J2EE clients hosted in WebSphere Application Server but required by the .NET server. The solution is to configure the binding of the J2EE client deployment descriptor to include both a nonce and a time stamp.

2. "CWSCJ0053E: Authorization failed for /UNAUTHENTICATED..."

This is a client/server authorization error associated with Enterprise Java Beans-based Web services where the access ID has been specified as *null*. This type of issue could be difficult to resolve because it is possible that it is propagated from a downstream resource that is not part of the EJB container that hosts the Web service. While the host machine's EJB container is satisfied with a null access ID, a non-null value is needed to satisfy the role requirements of the downstream server. This issue is resolved by configuring the security extensions of the EJB Web service to include the necessary credentials for the next participant in the request.

3. "WSEC6664E: Null is not allowed to PKIXBuilderParameters. The configuration of TrustAnchor and CertStoreList are not correct"

This is caused by an incorrectly specified certificate path in the application server and can be resolved through the administrative console. The WebSphere Information Center suggests the following:

- **In the administrative console, click Security -> Web services.**
- Under the Default consumer binding heading, click **Signing information -> *configuration_name***.
- Select either the **Trust any** or **Dedicated signing information** option. If you select the **Dedicated signing information** option, select both a trust anchor and a certificate store from the configurations that are provided in the drop-down lists.

- Click **OK** and **Save** to the master configuration.

These Web service security errors are a sample of the various security issues that may be encountered.

Instructor notes:

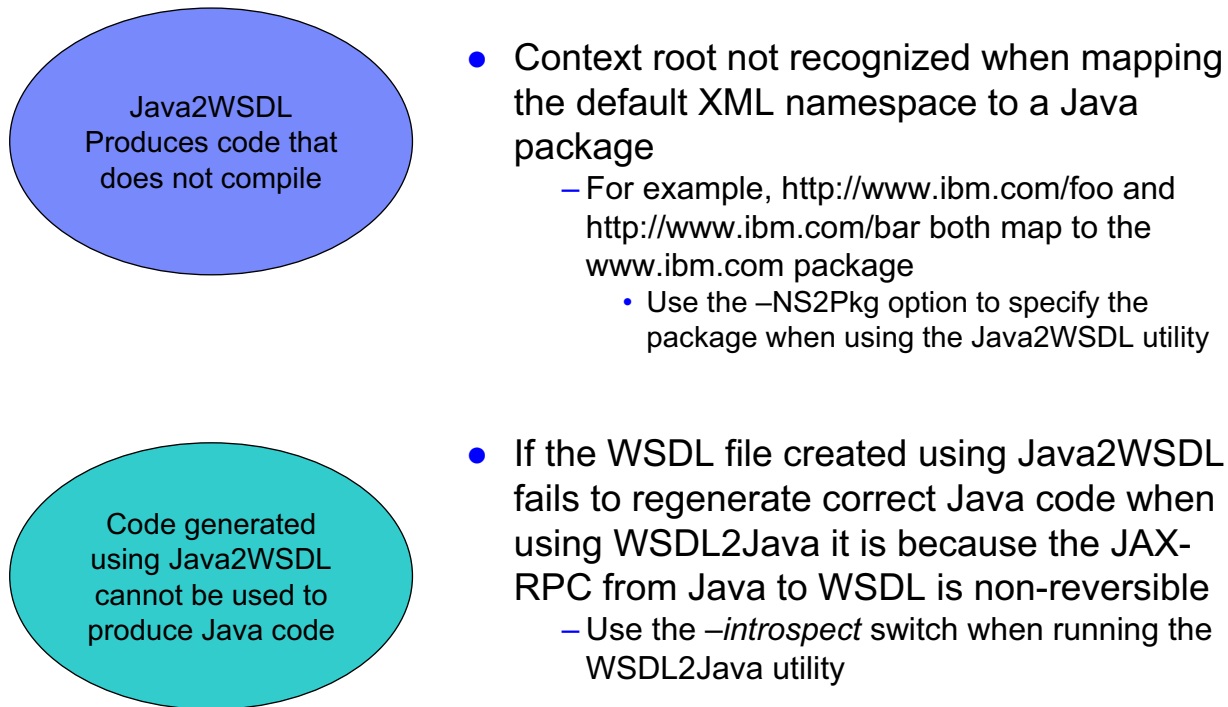
Purpose —

Details —

Additional information —

Transition statement —

Web service code generation problems



© Copyright IBM Corporation 2007

Figure 16-14. Web service code generation problems

WA5711.0

Notes:

Other problems encountered during code generation

Multiprotocol port component restrictions with JSR109 Version 1.0 and 1.1

A WSDL file containing http, EJB, and JMS bindings generated using the WSDL2Java command line utility could fail to deploy. This is because the Java Specification Requests (JSR) 109 requires each port component in the `web-services.xml` file to refer to a unique `servlet-class` element (JavaBean-based Web service) or `session` element (EJB-based Web service) in the `web.xml` file. If the Web service has multiple bindings, where each binding has multiple ports, the `web-services.xml` will have multiple port components where each point to a single EJB (session element) or JavaBean (servlet-class element). As a result, the `web.xml` or `ejb-jar.xml` will have an entry for each port with each referring to the same class. This does not conform to the JSR 109 specification and will result in an error.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Web service binding problems

The Enterprise
JavaBean (EJB)
fails to instantiate
when called

The WSDL file can
include specific
bindings by using
the `-bindingType`
option of the
Java2WSDL utility

- The error WSWS3422E: Error: Can not instantiate bean_name might signify that an EJB based Web service was called by client configured to call servlet based Web services
 - This error might be a result of improperly specified bindings in the WSDL file where a session bean is being accessed as a servlet based Web service
 - Work with the development team to synchronize the type of Web service that the developer intended to create
- For example; `java2wsdl -bindingTypes http,ejb -implClass my.pkg.MyEJBClass my.pkg.MySEI`

© Copyright IBM Corporation 2007

Figure 16-15. Web service binding problems

WA5711.0

Notes:

Enterprise software development, deployment, and publishing is a complex activity that usually involves multiple teams that could be located in multiple geographies. Good software practices with clear communication will resolve many problems. For example, binding problems should not occur if the WSDL file is stable and available to the various teams. However, during the development process it is typical for interfaces to be in a state of flux and only clear communication and change control processes will avoid these types of issues.

It is worth repeating the relationship between all of the stakeholders as identified by IBM's entry points. In addition to the technical resources responsible for realizing the Web service, there are business stakeholders that have to be considered. Even if the Web service functions without error, if it does not fulfill the business need that it was designed to address then it is of little use.

Instructor notes:

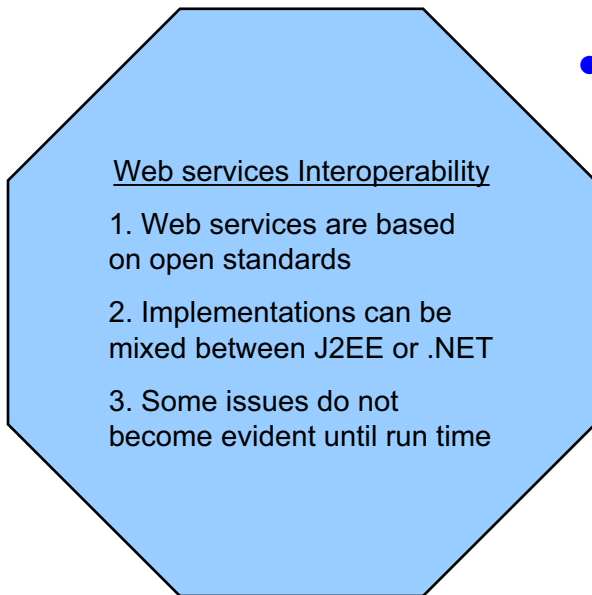
Purpose —

Details —

Additional information —

Transition statement —

Web service run time problems



- .NET client calling a J2EE Web service can result in an invalid operation exception
 - This is caused by a Web service that has a return type of `java.util.Vector` specified in a WSDL style based on `document/literal`

© Copyright IBM Corporation 2007

Figure 16-16. Web service run time problems

WA5711.0

Notes:

Some other problems encountered using command line tools

Error with the Endpoint Enabler tool on a hyperthreading-enabled computer

When developing Web services in an IBM tool such as Rational Application Developer, a race condition can exist between the automatic build and the Enterprise JavaBeans validator. Using the *Endpoint Enabler* on a computer that is *hyperthreading-enabled* might result in an error. To resolve this problem, disable automatic builds in the Rational Application Developer menu item **Project -> Build Automatically**. Once the changes are deployed, automatic builds can be reactivated.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. Name some of the command line tools available for generating Web services.
2. True or False: A Web service can be generated from a WSDL file but not from an EJB or JavaBean.
3. True or False: An EJB based Web service can be bound to, directly.
4. Discuss any problem that might occur between a .NET client and a J2EE Web service.

© Copyright IBM Corporation 2007

Figure 16-17. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1. Some of the command line tools available for generating Web services are:
 - a.
 - b.
2. True or false? Explain.
3. True or false? Explain.
4.
 - a.

Instructor notes:**Purpose —****Details — Answers:**

1. Java2WSDL and WSDL2Java are 2 of the most common tools. However, there are also tools for publishing to a UDDI registry as well as creating a Web service client.
2. False. A Web service can be generate from a WSDL file, and EJB, and a standard JavaBean.
3. True. The Web services standards provide direct binding to EJBs.
4. A problem can occur when using a .NET client with a J2EE Web service resulting in a *System.InvalidOperationException*. This is caused by a Web service that has a return type of *java.util.Vector* specified in a WSDL style based on *document/literal*.

Additional information —**Transition statement —**

Checkpoint solutions

1. Name some of the command line tools available for generating Web services.
 - **Java2WSDL and WSDL2Java are two of the most common tools. However, there are also tools for publishing to a UDDI registry as well as creating a Web service client.**
2. True or False: A Web service can be generated from a WSDL file but not from an EJB or JavaBean.
 - **False. A Web service can be generated from a WSDL file, an EJB, and a standard JavaBean.**
3. True or False: An EJB based Web service can be bound to, directly.
 - **True. The Web services standards provide direct binding to EJBs.**
4. Discuss any problem that might occur between a .NET client and a J2EE Web service.
 - **A problem can occur when using a .NET client with a J2EE Web service resulting in a *System.InvalidOperationException*. This is caused by a Web service that has a return type of *java.util.Vector* specified in a WSDL style based on *document/literal*.**

© Copyright IBM Corporation 2007

Figure 16-18. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic objectives

After completing this topic you should be able to:

- Use the errors and exceptions for tracking down further information
 - Leverage the IBM Support Assistant to search multiple knowledge bases
- Employ data collection and analysis tools to enable the appropriate trace string, reproduce the problem, and describe the problem
 - The WebSphere *MustGathers* for Web services contain the steps associated with debugging the problem
- Configure runtime tooling to inspect the Web service messages as they travel between the client and server
 - Use the *tcpmon* utility to inspect the run time SOAP messages between the client and server
 - Use Tivoli Performance Viewer to monitor run time performance

© Copyright IBM Corporation 2007

Figure 16-19. Topic objectives

WA5711.0

Notes:

As with any problem determination session, it is important to eliminate the easiest potential causes prior to doing a deep investigation of the problem.

It is common to begin by doing a search of available knowledge bases to determine if this is a known problem with a documented solution. The IBM Support Assistant can assist with searching multiple knowledge bases simultaneously. If a documented solution does not exist, then leverage the IBM Support facilities such as the *MustGathers*. If there is a guide for this particular problem it is extremely useful as a step-by-step process for gathering further information about the problem being experienced. As a part of the problem determination session, trace can be activated in the application server and as arguments to the JVM. Once the problem has been reproduced, collection tools can be used to gather the log, trace, and configuration files for a closer inspection. If the trace information does not lead to a resolution then there are runtime tools that can assist with inspecting the content and behavior of the Web service request/response between the client and server. The *tcpmon* utility can be used to inspect the SOAP messages—the content—that flows between the client and server and the Tivoli Performance Viewer can be activated to provide a picture of the runtime behavior of the system from a resource perspective.

Instructor notes:

Purpose —

Details —

Additional information —

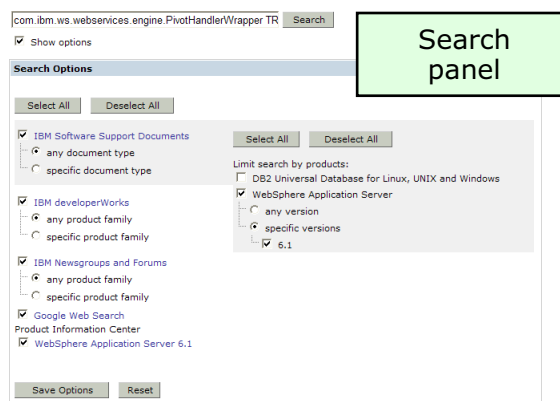
Transition statement —

Using the IBM Support Assistant Search

- Once an error is detected, the error code and explanation can serve as input into IBM Support Assistant search facility or the IBM Information Center search, directly
 - Consider the following exception:

```
com.ibm.ws.webservices.engine.PivotHandlerWrapper TRAS0014I: The following exception was
loggedjava.lang.NoSuchMethodError: com.ibm.wssvt.acme.websvcs.ExtWSPolicyData: method
getStartDate()Ljava/util/Date; not found at
com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.addElement(ExtWSPolicyData_Ser.java: 210) at
com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.serialize (ExtWSPolicyData_Ser.java:29) at
com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.serializeActual (SerializationContextImpl.java
719) at com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.serialize
(SerializationContextImpl.java: 463)
```

- The search steps are:
 - Input the search string
 - Specify the knowledge bases using **Select All**
 - Specify the product version
 - Click the **Search** button



© Copyright IBM Corporation 2007

Figure 16-20. Using the IBM Support Assistant Search

WA5711.0

Notes:

The IBM Support Assistant integrates knowledge bases, product information, data collection tools, and analysis tools.

The integrated search is the logical place to begin the problem determination session. The error messages, exceptions, or symptoms can be used as search criteria and the Search Options can confine the search to various sub-sets of knowledge bases and product versions. The exception dump includes:

- The class that threw the exception – `com.ibm.ws.webservices.engine.PivotHandlerWrapper`
- The message identifier associated with the error – `TRAS0014I` – which is composed of the prefix (`TRAS`), the number (`0014`), and the type of message (`I` = informational).
- The exception that was thrown – `java.lang.NoSuchMethodException`

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using the MustGathers

- There are several MustGathers that are focused on common Web services problems. The top level problem categories include:
 - Web services (for example: SOAP or UDDI or WSGW or WSIF)
 - Web services security
- It is important to keep in mind that a Web service is implemented using the standard J2EE facilities supported by WebSphere Application Server v6.1
 - Web services can be implemented using a dynamic Web application and run in the Web container, an EJB application and run in the EJB container, or a combination of both

© Copyright IBM Corporation 2007

Figure 16-21. Using the MustGathers

WA5711.0

Notes:

The WebSphere Application Server *MustGathers* are available from the WebSphere Application Server Support portal (<http://www.ibm.com/software/webservers/appserv/was/support/>) by clicking the link “MustGather: Read first.” The MustGathers are categorized by problem type with each type linked to a *MustGather* documents that are associated with the problem.

MustGather Web service problem categories

1. Web services (for example: SOAP or UDDI or WSGW or WSIF)
 - a. MustGather: Web services gateway problems in WebSphere Application Server V6.0, 5.1, and 5.0
 - b. MustGather: Problems with the Web Services Invocation Framework (WSIF) in WebSphere Application Server V6.0, 5.1 or 5.0
 - c. MustGather: Web services engine and tooling problems for WebSphere Application Server V6.1, V6, V5.1 and V5.

- d. Redpaper: WebSphere Application Server V6.1 Web Services Problem Determination

**Note**

While some of the MustGathers do not specifically cite WebSphere Application Server V6.1 in their title, they are still applicable to this version of the application server.

- 2. Web services security
 - a. MustGather: Web services security (WS-Security) problems with WebSphere Application Server

Instructor notes:

Purpose —

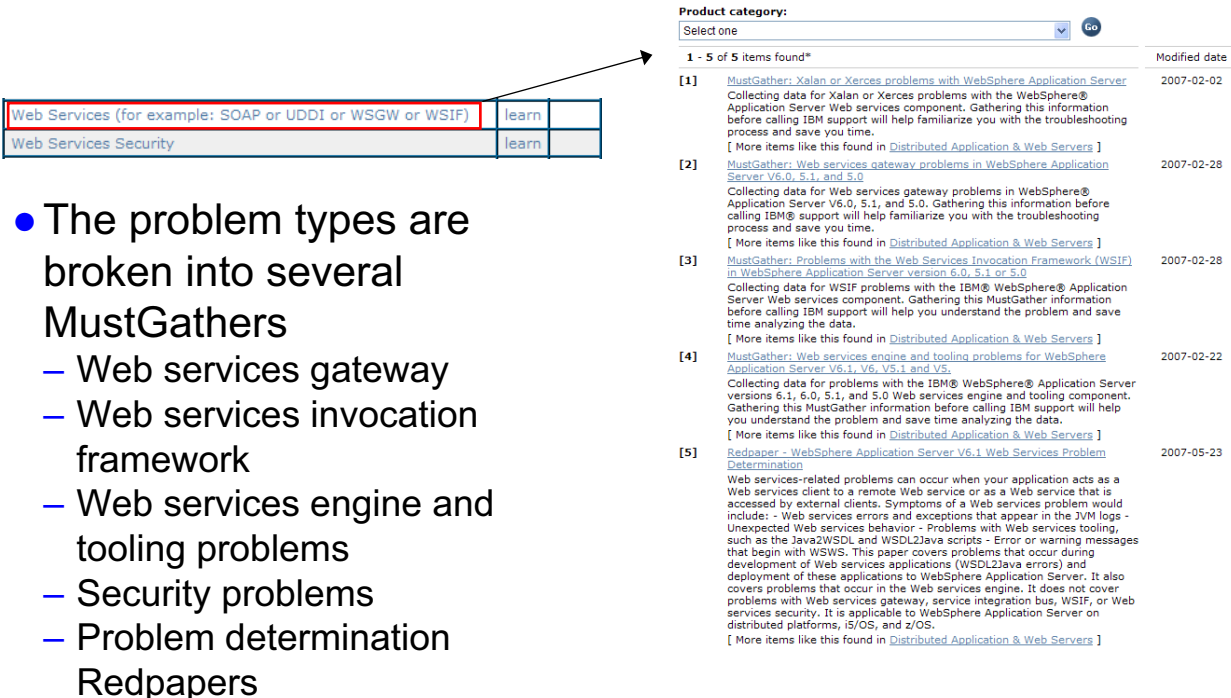
Details —

Additional information —

Transition statement —

Gathering the appropriate data

Web Services (for example: SOAP or UDDI or WSGW or WSIF)	learn	
Web Services Security	learn	



Product category:
Select one

1 - 5 of 5 items found*

		Modified date
[1]	MustGather: Xalan or Xerces problems with WebSphere Application Server Collecting data for Xalan or Xerces problems with the WebSphere® Application Server Web services component. Gathering this information before calling IBM support will help familiarize you with the troubleshooting process and save you time. [More items like this found in Distributed Application & Web Servers]	2007-02-02
[2]	MustGather: Web services gateway problems in WebSphere Application Server V6.0, 5.1, and 5.0 Collecting data for Web services gateway problems in WebSphere® Application Server V6.0, 5.1, and 5.0. Gathering this information before calling IBM® support will help familiarize you with the troubleshooting process and save you time. [More items like this found in Distributed Application & Web Servers]	2007-02-28
[3]	MustGather: Problems with the Web Services Invocation Framework (WSIF) in WebSphere Application Server version 6.0, 5.1, or 5.0 Collecting data for WSIF problems with the IBM® WebSphere® Application Server Web services component. Gathering this MustGather information before calling IBM support will help you understand the problem and save time analyzing the data. [More items like this found in Distributed Application & Web Servers]	2007-02-28
[4]	MustGather: Web services engine and tooling problems for WebSphere Application Server V6.1, V6, V5.1 and V6. Collecting data for problems with the IBM® WebSphere® Application Server versions 6.1, 6.0, 5.1, and 5.0 Web services engine and tooling component. Gathering this MustGather information before calling IBM support will help you understand the problem and save time analyzing the data. [More items like this found in Distributed Application & Web Servers]	2007-02-22
[5]	Redpaper - WebSphere Application Server V6.1 Web Services Problem Determination Web services-related problems can occur when your application acts as a Web services client to a remote Web service or a Web service that is accessed by external clients. Symptoms of a Web services problem would include: - Web services errors and exceptions that appear in the JVM logs - Unexpected Web services behavior - Problems with Web services tooling, such as the Java2WSDL and WSDL2Java scripts - Error or warning messages that begin with WSWS. This paper covers problems that occur during development of Web services applications (WSDL2Java errors) and deployment of these applications to WebSphere Application Server. It also covers problems that occur in the Web services engine. It does not cover problems with Web services gateway, service integration bus, WSIF, or Web services security. It is applicable to WebSphere Application Server on distributed platforms, i5/OS, and z/OS. [More items like this found in Distributed Application & Web Servers]	2007-05-23

- The problem types are broken into several MustGathers
 - Web services gateway
 - Web services invocation framework
 - Web services engine and tooling problems
 - Security problems
 - Problem determination Redpapers

© Copyright IBM Corporation 2007

Figure 16-22. Gathering the appropriate data

WA5711.0

Notes:

Prior to gathering the data it is useful to reproduce the problem with the appropriate trace levels set. The *MustGathers* or IBM Support can supply trace strings focused on the specific problem. Some of the common components involved in processing Web services include:

- com.ibm.ws.webservices
- com.ibm.ws.webservices.wssecurity
- com.ibm.ws.wssecurity
- com.ibm.xml
- com.ibm.xml.soapsec

Data can be gather manually or by using the IBM Support Assistant to automate the process. Some of the artifacts needed in order to determine the root cause of the problem include:

- Standard logs associated with server (SystemOut, SystemErr, native_stderr.log, native_stdout.log)
- Trace output
- The Web service WSDL, XML, and extension files
- A simple reproducible test case
- An understanding of the system topology

The *MustGathers* will provide step-by-step instructions for collecting the necessary information.

Instructor notes:

Purpose —

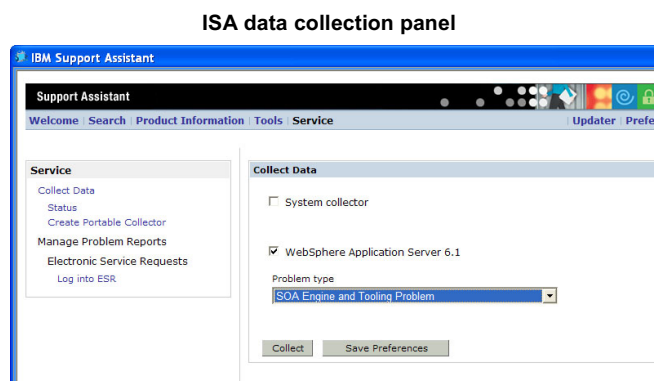
Details —

Additional information —

Transition statement —

Automating a Web service MustGather with the IBM Support Assistant

- Many of the Web services *MustGathers* are automated using the WebSphere Application Server v6.1 data collection plug-ins available through the IBM Support Assistant
 - Currently, the IBM Support Assistant has the following Web service scripts;
 - SOA Engine and Tooling Problem, SOA Security Problem, SOA Gateway Problem, and SOA UDDI Problem



© Copyright IBM Corporation 2007

Figure 16-23. Automating a Web service MustGather with the IBM Support Assistant

WA5711.0

Notes:

The steps involved in the *Web services engine and tooling problems for WebSphere Application Server V6.1, V6, V5.1 and V5* MustGathers are focused on assisting the user to navigate through the following areas:

To determine a test case that will verify when the issue is resolved:

- A test case is part of the exit criteria for knowing when the problem is solved by providing a way to test a hypothesis. The MustGather typically requests that such a case be defined prior to proceeding.
3. By supplying a checklist that results in a logical categorization of the problem:
- *MustGathers* include questions that focus on the facts about the problem by narrowing the problem domain to the set of circumstances that occur before, during, and after the problem occurred. An example of such questions include queries about when the system was stable, what changed in the system since the stable state, what the various components of the enterprise are, and how the system should behave when it is running correctly. Checklists are one of the most powerful problem determination tools

available because they assist in eliminating unnecessary *noise* in the problem determination process.

4. As a trace specification that is specific to the problem being experienced:
 - Setting the trace in WebSphere Application Server is difficult to determine in the context of a given problem. The MustGather guide will contain a trace specification that will be as minimally invasive as possible while optimized to generate enough trace to determine what the root cause of the problem might be.
5. By suggesting various tools or scripts that can assist in the problem determination process:
 - Both discovery and location of the appropriate tools is difficult to determine. The MustGather documentation not only suggest applicable tools but strives to include the necessary scripts and binaries to invoke the tooling. A good example of this is the Web services engine and tooling problems for WebSphere Application Server V6.1, V6, V5.1 and V5 MustGather document contains shell scripts for running the tcpmon tool to inspect the SOAP messages that flow between the client and server.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Setting the Web services specific trace strings

- A MustGather will typically include a trace string that can be set in the application server
 - General Web services trace string


```
com.ibm.ws.webservices.engine.*=all=enabled
```
 - Web services engine and tooling problem trace string


```
*=info:com.ibm.ws.webservices.*=all:HTTPChannel=all:GenericBNF=all
```

Excerpt from the Web service engine and tooling MustGather

Enabling Web service trace for WebSphere V6.1 and V6

- a. Start WebSphere Application Server.
- b. Open up the administrative console.
- c. Expand **Servers > Application Servers > server_name**.
- d. Select **Diagnostic Trace Service**.
- e. Set Maximum File Size to 200 MB and select File radio button in the General properties section. Set appropriate number of historical trace files.
- f. Navigate to **Change Log Detail Levels** in the **Additional properties** section.
- g. Clear the current trace string and add the following trace string to the General properties field for general Web services issues:


```
*=info:com.ibm.ws.webservices.*=all:HTTPChannel=all:GenericBNF=all
```
- h. Click **Apply** and **Save**.

© Copyright IBM Corporation 2007

Figure 16-24. Setting the Web services specific trace strings

WA5711.0

Notes:

The trace string can be set by using the *wsadmin* command line tool or the WebSphere Application Server v6.1 administrative console.

Setting trace using wsadmin and Jacl

1. Start the wsadmin client.

```
> wsadmin -profileName Dmgr01 -conntype SOAP -host myhostname -port 8880
-user myuser -password mypass
```

2. Get the server name by entering the command:

```
wsadmin>set server [$AdminConfig getid
/Cell:mycell/Node:mynode/Server:server1/]
```

which returns

```
server1(cells/mycell/nodes/mynode/servers/server1|server.xml#Server_1187
202397703)
```

3. Identify the trace service associated with the server.

```
wsadmin>set tc [$AdminConfig list TraceService $server]
```

which returns

```
(cells/mycell/nodes/mynode/servers/server1|server.xml#TraceService_11872  
02397703)
```

4. Set the trace.

```
wsadmin>$AdminConfig modify $tc {{startupTraceSpecification  
com.ibm.webservices.engine.*=all=enabled}}
```

5. Finally, save the configuration.

```
wsadmin>$AdminConfig save
```

The trace can also be set through the administrative console.

1. Log into the administration console.
2. From the menu, open **Troubleshooting** select **Logs and Trace**.
3. Select the server.
4. Choose **Log Level Details**.
5. Type the trace string, `com.ibm.webservices.engine.*=all=enabled`, in the text area.
6. Save the configuration by clicking the link save and the clicking the **OK** button.

It should be noted that trace can also be configured by navigating through the various components available in the Log Level Details panel by starting at **com.ibm.ws** and select **webservices**, **engine**, and **all**.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Activating trace and reproducing the problem

- The WebSphere Application Server v6.1 contains a highly granular tracing facility that can be configured to
- Whether data collection is manual or automated through the IBM Support Assistant, the analysis flow will generally follow a pattern such as:
 - Inspect the stream files to look for exceptions and stack traces
 - Look through the First Failure Data Capture logs to determine if there are runtime exceptions that are being caught prior to making it to the streams
 - Look at the configuration associated with the Web service, such as the web.xml, web-service.xml, ws-security.xml, the WSDL file, the ejb-jar.xml, and any properties files used by the applications
 - Inspect the cell, node, and server configuration files for values that are outside of best practices

© Copyright IBM Corporation 2007

Figure 16-25. Activating trace and reproducing the problem

WA5711.0

Notes:

The First Failure Data Capture (FFDC) facility is composed of 2 file formats, an **index file** and an **exception file**. For each server, there is an index file containing meta-information about each exception file. A new set of index files, for each JVM, are created when the JVM is restarted. A new exception file is created when the exception occurs.

FFDC index file

The index file has a naming convention of `<server_name>_exception.log`.

For example, `nodeagent_exception.log`.

The format of the index file is:

Index | Count | Time of last Occurrence | Exception | Sourceld | Probeld

1. **Index** – An integer that uniquely identifies the exception class and exception source class combination within this index file

- FFDC incurs very little overhead because it spools the log entries in memory until a pre-defined time span has elapsed. The index is unique within that time span.
2. **Count** – The number of times that the exception has been logged
 - Rather than creating an entry and an exception file for each occurrence, FFDC keeps a count of how many times this exception has occurred.
 3. **Time of last Occurrence** – The timestamp signifying the last time the exception occurred
 - Updated when the Count value is incremented
 4. **Exception** – The Java exception object that was thrown
 5. **SourceId** – The fully qualified class name that threw the exception
 6. **Probeld** – The ID of the probe that handled this exception

Here is an example index file entry:

```

Index  Count  Time of last Occurrence  Exception SourceId ProbeId
-----+-----+-----+-----+-----+-----
1      2      7/19/07 15:29:47:359 CDT java.net.SocketTimeoutException
com.ibm.ws.management.component.JMXConnectors.MulticastVerifier.run
979
2      3      7/19/07 15:33:18:640 CDT java.io.IOException
com.ibm.ws.management.discovery.DiscoveryService.sendQuery 165
3      3      7/19/07 15:33:18:640 CDT java.net.SocketException
com.ibm.ws.management.discovery.transport.MUdpMessenger.openSocket
134
4      1      7/19/07 15:29:17:281 CDT java.net.SocketException
com.ibm.ws.management.component.JMXConnectors.MulticastVerifier 959
5      1      7/19/07 15:30:31:546 CDT
java.lang.reflect.InvocationTargetException
com.ibm.websphere.management.AdminClientFactory.createAdminClient 215
6      2      7/19/07 15:30:31:546 CDT
com.ibm.websphere.management.exception.ConnectorNotAvailableException
com.ibm.ws.management.connector.soap.SOAPConnectorClient.reconnect
269

```

From this example, the exception

jcom.ibm.websphere.management.exception.ConnectorNotAvailableException
thrown from the class

com.ibm.ws.management.connector.soap.SOAPConnectorClient.reconnect has a unique ID of 6, has occurred two times since the server was last started, and was captured by probe ID 269.

FFDC exception file

An exception file is created for each unique id within the index file. An example of an exception file is:

```
-----Start of DE processing----- = [7/25/07 15:25:20:828 CDT] , key =
com.ibm.websphere.management.exception.ConnectorNotAvailableException
com.ibm.ws.management.connector.soap.SOAPConnectorClient.reconnect 269
Exception =
com.ibm.websphere.management.exception.ConnectorNotAvailableException
Source = com.ibm.ws.management.connector.soap.SOAPConnectorClient.reconnect
probeid = 269
Stack Dump =
com.ibm.websphere.management.exception.ConnectorNotAvailableException:
ADMC0016E: The system cannot create a SOAP connector to connect to host
rallan-lt60p at port 8885.
    at
    com.ibm.ws.management.connector.soap.SOAPConnectorClient.getUrl (SOAPConn
ectorClient.java:1102)
    at
    com.ibm.ws.management.connector.soap.SOAPConnectorClient.access$300 (SOAP
ConnectorClient.java:108)
    at
    com.ibm.ws.management.connector.soap.SOAPConnectorClient$4.run (SOAPConne
ctorClient.java:303)
    at
    com.ibm.ws.security.util.AccessController.doPrivileged (AccessController.
java:118)
    .
    .
    .
Caused by: java.net.SocketException: Operation timed out: connect:could be
due to invalid address
    at java.net.PlainSocketImpl.socketConnect (Native Method)
    at java.net.PlainSocketImpl.doConnect (PlainSocketImpl.java:372)
    .
    .
    .
Dump of callerThis =
Object type = com.ibm.ws.management.connector.soap.SOAPConnectorClient
com.ibm.ws.management.connector.soap.SOAPConnectorClient@7db67db6
==> Performing default dump from com.ibm.ws.management.dm.ConnectorDM = Wed
Jul 25 15:25:21 CDT 2007
```

```
SOAP Connector client properties: = {port=8885, securityEnabled=true,  
type=SOAP, host=rallan-lt60p, isInternal=true}+Data for directive  
[defaultconnector] obtained. =  
==> Dump complete for com.ibm.ws.management.dm.ConnectorDM = Wed Jul 25  
15:25:21 CDT 2007
```

This exception file contains the exception stack trace, the fully qualified class name of the class that threw the exception, the probe ID, and port, host, and security information.

FFDC can be extremely useful for determining the root cause of transient problems that do not cause a complete failure and are handled prior to being logged in the system error log files.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Configuring tcpmon from the command line

- The tcpmon tool is available as a command line utility or as an embedded tool in the IBM Rational Application Developer workbench.
 - It is useful for inspecting the contents of SOAP messages as they travel between the client and server
 - The client sends SOAP request to this unique port and tcpmon unwraps the message and forwards it on to the actual Web service
 - The Web service responds to tcpmon which displays the response and responds to the client

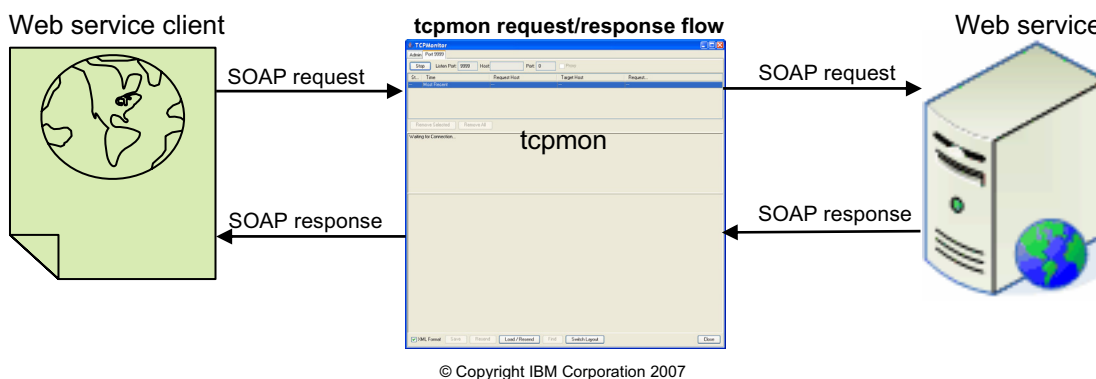


Figure 16-26. Configuring tcpmon from the command line

WA5711.0

Notes:

The *tcpmon* utility is setup to listen on a unique port and act as a proxy between the Web service client and the Web service. *tcpmon* displays the SOAP request and response envelopes that flow back and forth between the Web services client and the Web service.

The `com.ibm.ws.admin.client_6.1.0.jar` and `com.ibm.ws.webservices.thinclient_6.1.0.jar` archives contain the classes that implement *tcpmon*.

To run *tcpmon* from the command line do the following:

1. Run the **setupCmdLine.bat** (Windows) or the **setupCmdLine.sh** (UNIX) command from the `<profile_root>/<application_server>/bin` directory
 - a. Set or export the path variables `set PATH=%WAS_PATH%;%PATH%` (Windows) or `export PATH=$WAS_PATH:$PATH` (UNIX).
 - b. Add the jar files that implement *tcpmon* to the environment.
 - c. Run the **java -Djava.ext.dirs=%WAS_EXT_DIRS% com.ibm.ws.webservices.engine.utils.tcpmon** command

2. After the tcpmon window appears, click the **Add** button and create a listener on a unique port. For example, if the Web service is listening on port 9080, set the tcpmon listen port to 9888.

Instructor notes:

Purpose —

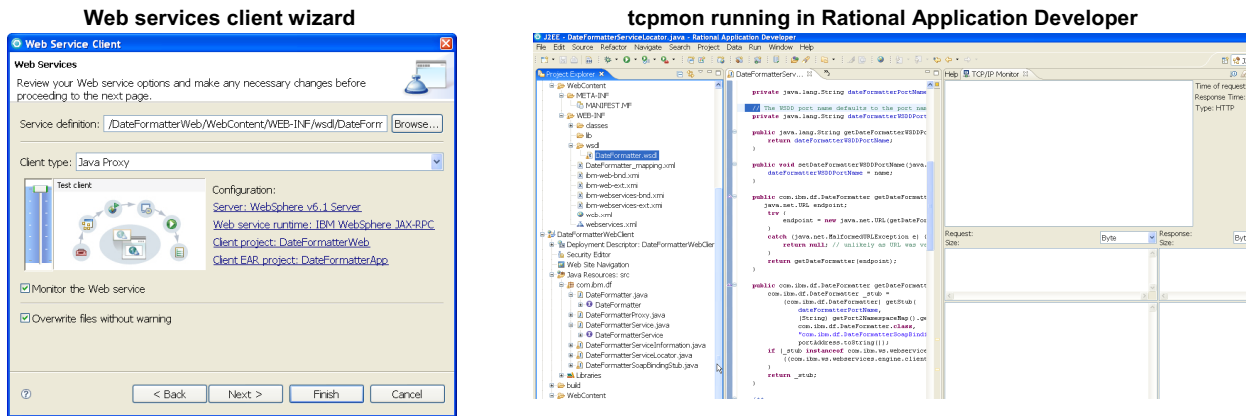
Details —

Additional information —

Transition statement —

Running tcpmon from IBM Rational Application Developer

- A Web service client can be generated in the IBM Rational Application Developer environment and configured to monitor the SOAP messages using the tcpmon utility



© Copyright IBM Corporation 2007

Figure 16-27. Running tcpmon from IBM Rational Application Developer

WA5711.0

Notes:

To run an integrated version of tcpmon inside of Rational Application Developer, do the following:

1. In Rational Application Developer, import the Web service and use the WSDL file of the Web service to generate a Web service client.
2. Once the Web service has been imported, right-click the WSDL file to highlight it.
3. Select **New -> Project -> Other -> Web Service -> Web Services Client**.
4. Set the client slider to **Test Client** and check the **Monitor Web Service** check box.

Once the client has been generated it can be driven by a Web browser and the SOAP messages will be displayed in the RAD console.

Instructor notes:

Purpose —

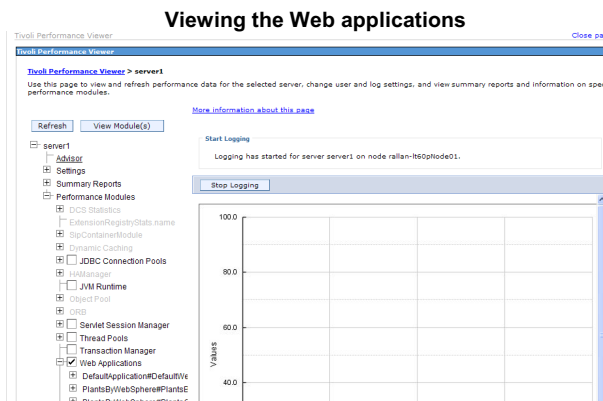
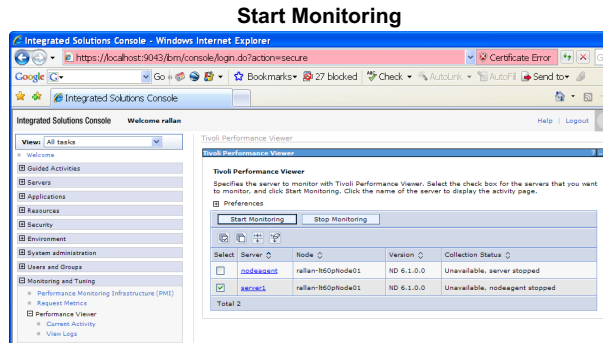
Details —

Additional information —

Transition statement —

Using the Tivoli Performance Viewer to monitor performance problems in Web services

- The Tivoli Performance Viewer is integrated into the WebSphere administrative console
 - The Tivoli Performance Monitor reads the metrics that are emitted from the Performance Management Interface (PMI)
- A real-time report is generated that is useful for locating performance bottlenecks
 - Questions regarding memory, threads, connections, and other runtime resource consumption can be answered by monitoring the Web service during run time
- Any time the PMI is activated there is a performance penalty as a result
 - A good strategy is to activate the PMI on a subset of the nodes, only



© Copyright IBM Corporation 2007

Figure 16-28. Using the Tivoli Performance Viewer to monitor performance problems in Web services

WA5711.0

Notes:

The data captured by the Tivoli Performance Monitor can be saved for later analysis. The size of the data file, containing the metrics, will grow very quickly so it is important to use monitoring sparingly, targeting runtime behaviors that best represent the performance issue trying to be resolved.

Using the Tivoli Performance Viewer requires configuring the Performance Monitoring Infrastructure (PMI).

1. From the administrative console menu, select the Monitoring and Tuning and Performance Monitoring Infrastructure (PMI) option.
2. Click the server to be monitored.
3. Select the monitoring level.
4. Click **OK**.
5. Click **Save** to save the changes.
6. Configuring the modules to display:

- a. From the administrative console menu, select Monitoring and Tuning, Performance Viewer, and Current Activity.
- b. Select the monitored server.
- c. Expand the Performance Modules branch of the server tree.
- d. Select the modules to be monitored.
- e. Click the **Start Logging** button.

By clicking the **View Modules** button a graphic or tabular display is available. The graphical display relies on the Adobe SVGViewer which might need to be downloaded on first use. If this is the case, the download page will be displayed in the current administrative panel and the installation of the viewer can take place without leaving the administrative console. Once completed, a graphical view of the metrics will be available for display.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. What set of documents does IBM provide for stepping through a problem determination session?
2. Discuss some of the information you would find in these documents
3. What are the various log files that can be inspected in order to find exceptions?
4. What is the utility that can be used to view SOAP messages as they travel between the Web service client and Web service?

© Copyright IBM Corporation 2007

Figure 16-29. Checkpoint

WA5711.0

Notes:

Write down your answers here:

- 1.
2.
 - a.
 - b.
 - c.
3.
 - a.
 - b.
 - c.
- 4.

Instructor notes:

Purpose —

Details — Answers:

1. *These documents are called... The MustGathers*
2. *Some of the information in these documents include:*
 - a. *The trace strings*
 - b. *Instructions on reproducing the problem*
 - c. *The information to collect*
3. *Some of the log files that include exceptions are:*
 - a. *The SystemErr.log*
 - b. *The trace.log*
 - c. *The First Failure Data Capture or FFDC logs*
4. *The utility that can be used to view messages that travel between the Web service client and Web services is... tcpmon*

Additional information —

Transition statement —

Checkpoint solutions

1. What set of documents does IBM provide for stepping through a problem determination session?
 - **These documents are called MustGathers.**
2. Discuss some of the information you would find in these documents
 - **Some of the information in these documents include:**
 - **The trace strings**
 - **Instructions on reproducing the problem**
 - **The information to collect**
3. What are the various log files that can be inspected in order to find exceptions?
 - **Some of the log files that include exceptions are:**
 - **The SystemErr.log**
 - **The trace.log**
 - **The First Failure Data Capture or FFDC logs**
4. What is the utility that can be used to view SOAP messages as they travel between the Web service client and Web service?
 - **The utility is tcpmon**

© Copyright IBM Corporation 2007

Figure 16-30. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Topic objectives

- After completing this topic you should be able to:
- Use the IBM development environments to resolve various problems such as:
 - Authentication issues in a secure environment
 - Interoperability problems caused by incorrect configuration
 - Client runtime problems
- Use the IBM deployment tools to re-deploy the Web service to IBM WebSphere Application Server v6.1
 - The IBM Rational Application Developer workbench and the IBM Application Server Toolkit provide facilities for importing, modifying, validating, and exporting a Web service or Web service client
 - In addition, Web services and Web service clients can be tested in these environments prior to re-deployment
- Use the IBM WebSphere Application Server v6.1 update wizard to re-deploy the Web service

© Copyright IBM Corporation 2007

Figure 16-31. Topic objectives

WA5711.0

Notes:

The Rational Application Developer workbench and the IBM Application Server Toolkit are similar in their appearance. The enterprise application that contains the Web service can be loaded into these tools and edited, tested, and repackaged for deployment to the application server. Rational Application Developer and IBM Application Server Toolkit support various perspectives including a J2EE project perspective, a Web application perspective, a Java perspective, and XML, WSDL, and properties file editors that assist with the problem resolution process.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Using RAD and AST to manage the Web service

- RAD and AST are Eclipse-based tools with full J2EE support
 - The import feature of the workbench can be used to open the Enterprise Application (EAR) file containing the Web service
 - Once the application is open in the workbench, the various artifacts can be viewed, modified, and validated before the application is exported

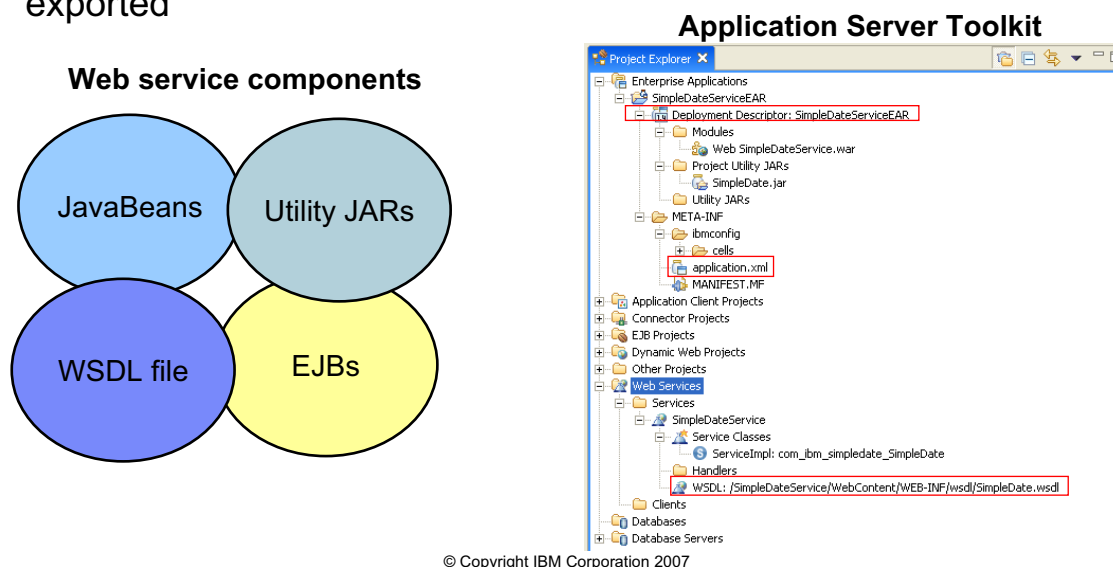


Figure 16-32. Using RAD and AST to manage the Web service

WA5711.0

Notes:

Depending on the client type, various proxy and WSDL element classes might exist. For example, classes that implement the binding, locator, and transport functionality will be needed to handle the client/server interaction. In order to resolve Web service problems an understanding of the relationship between the Web service artifacts is essential. The predominant Web service configuration and extension files include:

- The WSDL file – This is the contract that is implemented by any of the components participating in the Web service transaction. The WSDL file describes the service, the operations provided by the service, the arguments and return type of those operations, the location of the service, the communication protocols used to interact with the service, and the data types used by the service. The WSDL document is an XML document that adheres to the standard specified by w3.org.
- The webservices.xml file – This is the mapping between the Web service artifacts and includes a reference to the Web service description, WSDL file, the service endpoint interface, and the servlet that routes Web service requests.

- The `ibm-webservices-bnd.xmi/ibm-webservicesclient-bnd.xmi` file – This deployment descriptor contains information that is WebSphere specific or outside of the Java 2 Web service standard. This file includes description and port component links to the `webservices.xml` file and scoping information. If the Web service is secure, the key locator information will be included in this file.
- The `ibm-webservices-ext.xmi/ibm-webservicesclient-ext.xmi` file – This deployment descriptor is similar to the `ext` file but contains security information when SOAP security is enabled. SOAP security can consist of digital signature, XML digest, or both.
- The `<service_name>_mapping.xml` file – This file is a JAX-RPC mapping file that contains associations between the service, service endpoint, operation, and other elements of the WSDL file.
- The `<service_name>_SEI` Java class – The Service Endpoint Interface class is used by JAX-RPC to simplify the complexity of the binding and transport mechanism used in Web service transactions. The client and server provide implementations based on this interface and the complexity of the calling and receiving of the SOAP messages is handled by JAX-RPC.

Instructor notes:

Purpose —

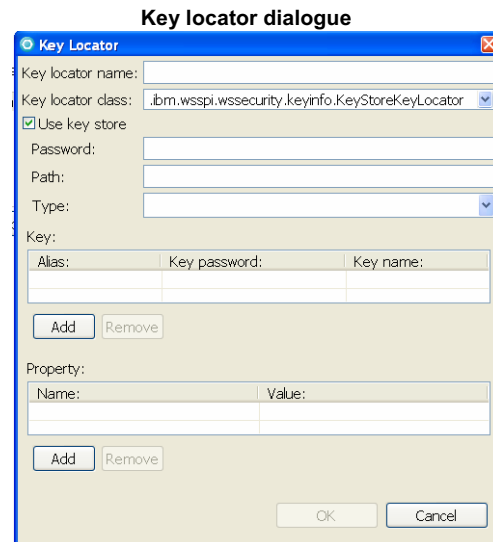
Details —

Additional information —

Transition statement —

Resolving authentication issues in a secure environment

- A failure between a Web service client and a Web service in a secure environment can be caused by incorrect credentials or certificate configuration
 - The KeyLocators element contains the path to the key store
 - Modify this element to correctly point to the key store will resolve this problem
 - The KeyLocators element can be in one of the following bind files
 - ws-security.xml
 - ibm-webservices-bnd.xmi
 - ibm-webservicesclient-bnd.xml
 - The Application Server Toolkit workbench can be used to modify the key store and credentials



© Copyright IBM Corporation 2007

Figure 16-33. Resolving authentication issues in a secure environment

WA5711.0

Notes:

The Web service bindings can be edited by opening the `ibm-webservice-bnd.xmi` file of the Web module using the Web services editor view of the RAD workbench. The *Key Locator* dialog panel contains the input fields for modifying the keystore password, path, and type values. If a secure Web service fails to start an exception of this type might occur:

```
[6/19/03 11:13:02:976 EDT] 421fdaa2 KeyStoreKeyLo E WSEC5156E: An exception
while retrieving the key from KeyStore object:
java.security.UnrecoverableKeyException: Given final block not properly
padded
```

The solution to this problem is to provide the correct values for the key locator element of the Web service. The values to verify are the password, path, and type of the key store used for the certificate exchange between the client and server.

An `ibm-webservices-bnd.xmi` file without the key locator information has the form:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<com.ibm.etools.webservice.wsbind:WSBinding xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:com.ibm.etools.webservice.wsbind="http://www.ibm.com/websphere/appserv
er/schemas/5.0.2/wsbind.xmi" xmi:id="WSBinding_1186325514765">
  <wsdescBindings xmi:id="WSDescBinding_1186325514765"
wsDescNameLink="DateFormatterService">
    <pcBindings xmi:id="PCBinding_1186325514765"
pcNameLink="DateFormatter"/>
  </wsdescBindings>
</com.ibm.etools.webservice.wsbind:WSBinding>

```

Once the key locator information is added the `ibm-webservice-bnd.xmi` file contains the `keyLocator` element:

```

<?xml version="1.0" encoding="UTF-8"?>
<com.ibm.etools.webservice.wsbind:WSBinding xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:com.ibm.etools.webservice.wsbind="http://www.ibm.com/websphere/appserv
er/schemas/5.0.2/wsbind.xmi" xmi:id="WSBinding_1186325514765">
  <wsdescBindings xmi:id="WSDescBinding_1186325514765"
wsDescNameLink="DateFormatterService">
    <pcBindings xmi:id="PCBinding_1186325514765" pcNameLink="DateFormatter">
      <securityRequestConsumerBindingConfig
xmi:id="SecurityRequestConsumerBindingConfig_1186498416875">
        <keyLocator xmi:id="KeyLocator_1186498416875" name="ExampleKeyLocator"
classname="com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator">
          <keyStore xmi:id="KeyStore_1186498416875"
storepass="{xor}Oic+Mi8zOi8+LCwoMC07" path="C:\Student\example.jks"
type="JKS"/>
          <keys xmi:id="Key_1186498416875" alias="keyAlias"
keypass="{xor}PjM2PiwPPiwsKDAto==" name="aliasName"/>
        </keyLocator>
      </securityRequestConsumerBindingConfig>
    </pcBindings>
  </wsdescBindings>
</com.ibm.etools.webservice.wsbind:WSBinding>

```

The `keyLocator` element includes information about the location, type, and password of the key store as well as the name and password of the key used in the authentication process. In a test scenario, where the client and server are on the same host machine, the values of the `keyLocator` element can be the same. However, in a production environment the location of the key store will depend on the directory structure and organization of the host machine.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Resolving interoperability problems caused by incorrect configuration

- Even though Web services are a standards based technology, the differences in vendor implementations can result in communication failure
 - A good example of this type of problem is the difference between how a vector object is handled in a .NET client and a WebSphere Web service
 - There is no literal translation of a vector object from the perspective of a .NET client
 - Avoid the use of vector objects in WebSphere Web services
 - One technique is to use a flattened string to pass multiple key/value pairs back and forth between the client and server
- A Web service client uses the value of the endpoint element, in the WSDL file, as the request URL
 - A request failure can be caused by an incorrect endpoint value
 - Use the IBM Rational Application Developer workbench to modify the WSDL file endpoint element so that it references the correct URL

© Copyright IBM Corporation 2007

Figure 16-34. Resolving interoperability problems caused by incorrect configuration

WA5711.0

Notes:

Another problem that can occur between a Web service client and Web service is in the serialization/deserialization of various data types when using Java 2 and the Java for XML Remote Procedure call (JAX-RPC) specification. For example, a `java.util.Date` object serialized as an XML `dateTime` data type cannot be deserialized to a `java.util.Date` object. It will be deserialized as a `java.util.Calendar` object on the server side. If the server and client share the same bindings, such as the bindings in a shared jar, then this problem will occur. The solution is to create separate bindings for the client and server deployment that resolve to the correct object type for the `xsd:dateTime` *type* attribute of the message part.

From the Product Information Center you get the following example exception:

```
com.ibm.ws.webservices.engine.PivotHandlerWrapper TRAS0014I: The following
exception was logged java.lang.NoSuchMethodError:
com.ibm.wssvt.acme.websvcs.ExtWSPolicyData: method
getStartDate()Ljava/util/Date;
not found at
```

```
com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.addElements (ExtWSPolicyData_Ser.java: 210) at
com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.serialize
(ExtWSPolicyData_Ser.java:29) at
com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.serializeActual (SerializationContextImpl.java 719) at
com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.serialize
(SerializationContextImpl.java: 463)
```

Given a WSDL file with the following signature:

```
<wsdl:message name="getDateResponse">
  <wsdl:part name="getDateReturn" type="xsd:dateTime"/>
</wsdl:message>
<wsdl:message name="getCalendarResponse">
  <wsdl:part name="getCalendarReturn" type="xsd:dateTime"/>
</wsdl:message>
```

Using the Java2WSDL utility to generate the interface will result in the following interface:

```
package server;
public interface Test_SEI extends java.rmi.Remote {
  public java.util.Calendar getCalendar () throws java.rmi.RemoteException;
  public java.util.Date getDate() throws java.rmi.RemoteException;
}
```

This will work for the Web service client; however, for the Web service server side interface the return type for the *getDate* method need to be changed to return a *java.util.Calendar* object to map to the deserialized object:

```
package server;
public interface Test_SEI extends java.rmi.Remote {
  public java.util.Calendar getCalendar () throws java.rmi.RemoteException;
  public java.util. Calendar getDate() throws java.rmi.RemoteException;
}
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Resolving client run time problems

- A WebServicesFault can occur in a clustered environment configured to leverage session persistence
 - This is resolved by setting the JVM property `com.ibm.websphere.webservices.http.requestResendEnabled=true`
 - Various JVM properties can be set on a Web service client using RAD, AST, or through the administrative console of the application server
 - In the RAD or AST environment, open the deployment descriptor of the client and set the key/value pair through the Web Services Client Bindings panel
 - In the WebSphere Application Server v6.1 administrative console navigate to **Servers -> Application Servers -> server -> Java and Process Management -> Process Definition -> Java Virtual Machine -> Custom Properties** and set the key/value pair
- Performance is impacted due to DNS resolution time
 - This type of problem is easily solved updating the machines HOSTS file to negate the necessity for contacting the DNS server

© Copyright IBM Corporation 2007

Figure 16-35. Resolving client run time problems

WA5711.0

Notes:

Some client/server problems are not specific to Web services and could be due to:

- Stopped or hung application server
- Incorrect timeout values in a intermediary between the client and server that cause connections to be dropped
 - For example, a firewall, proxy, and so on.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exporting the modified Web service

- Any of the IBM workbenches, such as IBM Rational Application Developer or the Application Assembly Toolkit, can be used to compile, package and export the Web service
 - Change the perspective to J2EE, select the enterprise application and choose **File -> Export**
 - The EAR file will be available for redeployment to IBM WebSphere Application Server v6.1

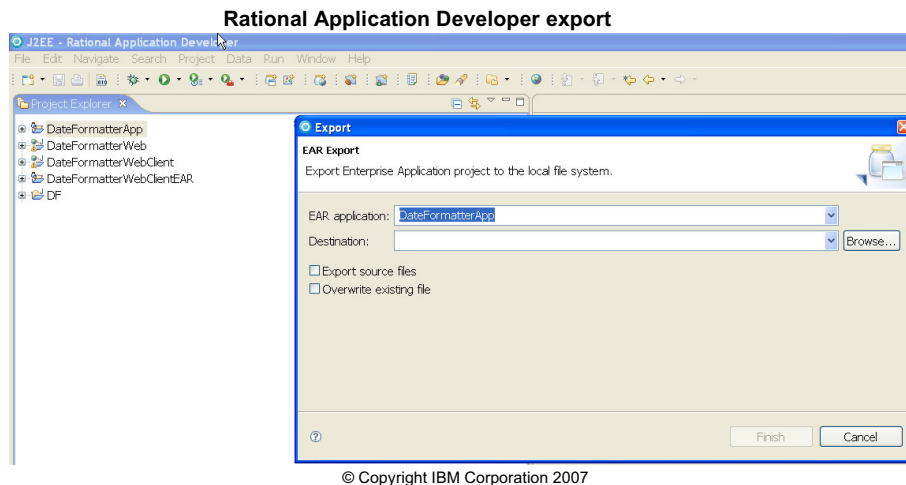


Figure 16-36. Exporting the modified Web service

WA5711.0

Notes:

For some problems, changes can be made to the Web service through the WebSphere administration console. The administrative console provides access to the Web service parameters while it is installed and running.

1. From the administrative console menu, select **Applications and Enterprise Applications**.
2. From the enterprise applications panel select the application of interest.

The many parameters that compose the Web service can be accessed, modified and saved from this view of the panel. For example, security roles can be modified, virtual hosts changed, Web service reload intervals, and EJB specific parameters, to name a few.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Redeploying the Web service

- The modified Web service is now ready to be redeployed to the application server
 - Log into the administrative console of IBM WebSphere Application Server v6.1
 - Navigate to **Applications -> Enterprise Applications**, select the Web service and click the **Update** button
 - The exported enterprise application can be selected from the file system and updated on the application server

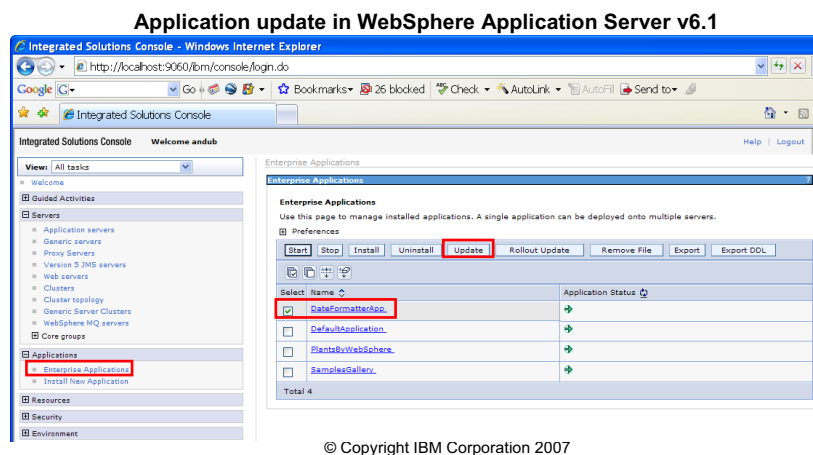


Figure 16-37. Redeploying the Web service

WA5711.0

Notes:

After the update process is complete, the updated application will be started and ready to receive requests. The test case can be run to determine if the problem has been resolved, and, to verify that functionality has not been regressed during the problem resolution activity. Besides running the repeatable test case the redeployment will provide an opportunity to determine if the runtime conditions that originally uncovered the Web service no longer occur. Close inspection of the Web service behavior and the logs is important to determine if a correct resolution has truly been achieved.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint

1. Name some of the artifacts (XML files and modules) associated with Web services.
2. True or False: The endpoint element in a Web service WSDL file is the Web service URL.
3. What is a potential cause of Web service performance problems?
4. True or False: Once a problem has been resolved in a Web service, using the Application Server Toolkit, the old application must be totally uninstalled from WebSphere Application Server V6.1 and re-installed.

© Copyright IBM Corporation 2007

Figure 16-38. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1.

Artifacts:

- a.
- b.
- c.

Components:

- a.
- b.

2. True or false? Explain.

3.

4. True or false? Explain.

Instructor notes:**Purpose —****Details — Answers:**

1. Some of the Web services artifacts and components are...

Artifacts:

- webservices.xml
- The WSDL file
- The Web service bind and extension files

Components:

- a. A WAR file containing the dynamic Web application and Web service
 - b. An EJB JAR file containing the EJB-based Web service
2. True. The endpoint element contains the URL of the Web service.
 3. A potential cause of Web service performance problems is repeated DNS resolutions for the endpoint URL.
 4. False. The *update* feature of WebSphere Application Server V6.1 can be leveraged to update the existing application.

Additional information —**Transition statement —**

Checkpoint solutions

1. Name some of the artifacts (XML files and modules) associated with Web services.
 - **Some of the Web services artifacts and components are:**
 - **Artifacts:**
 - webservices.xml
 - The WSDL file
 - The Web service bind and extension files
 - **Components:**
 - A WAR file containing the dynamic Web application and Web service
 - An EJB JAR file containing the EJB based Web service
2. True or False: The endpoint element in a Web service WSDL file is the Web service URL.
 - **True. The endpoint element contains the URL of the Web service.**
3. What is a potential cause of Web service performance problems?
 - **A potential cause of Web service performance problems is repeated DNS resolutions for the endpoint URL.**
4. True or False: Once a problem has been resolved in a Web service, using the Application Server Toolkit, the old application must be totally uninstalled from WebSphere Application Server V6.1 and re-installed.
 - **False. The update feature of WebSphere Application Server V6.1 can be leveraged to update the existing application.**

© Copyright IBM Corporation 2007

Figure 16-39. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

References

- New to SOA and Web services
 - New to SOA: <http://www.ibm.com/developerworks/webservices/newto/>
 - SOA Governance:
http://www.ibm.com/software/solutions/soa/entrypoints/advancing_soa_governance.html
- WebSphere Application Server V6.1 Web Services Problem Determination
 - <http://www.redbooks.ibm.com/redpapers/pdfs/redp4306.pdf>
- Troubleshooting Web service compiler problems
 - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rwbs_trbjavacompiler.html
- Web service command line tools trouble shooting tips
 - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rwbs_trbcommand.html
- Specifications
 - WSDL: <http://www.w3.org/TR/wsdl>

© Copyright IBM Corporation 2007

Figure 16-40. References

WA5711.0

Notes:

Download and install the IBM Support Assistant

1. Go to <http://www.ibm.com/software/support/isa>
2. Select IBM Support Assistant V3.
3. Download the installation package for any platform where you will install IBM Support Assistant.
 - You will need to log in using your IBM Web identity (same ID as used for My Support, developerWorks, and so forth).
 - If you do not already have an IBM Web identity, you may complete the free registration process to obtain one.
4. Download the compressed archive files.
5. Expand the archive file in a temporary directory using a tool such as WinZip.

The archive contains an installer program and the Installation and Troubleshooting Guide. Follow the directions in the Installation and Troubleshooting Guide to install IBM Support Assistant.

Instructor notes:

Purpose —

Details — The newest version of ISA (v3.0.2) has an improved updater. However, on slow network connections it is recommended to download one plug-in at a time. In the most severe case, while teaching this class, it might be necessary to use a local update site to perform the labs. Contact the ISA team to request such a site.

Additional information —

Transition statement —

Unit summary

Now that you have completed this unit, you should be able to:

- Understand Web services from a high level
 - SOA and Web services
 - The components and activities that make up Web services
 - Tools available for generating Web services
- Identify the common problems associated with developing and deploying Web services
 - Development problems
 - Deployment problems
 - Runtime problems
- Determine the root cause of Web services problems and resolve them
 - Code generation and compiler problems
 - Binding problems
 - Protocol problems

© Copyright IBM Corporation 2007

Figure 16-41. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 11: Troubleshooting Web services

© Copyright IBM Corporation 2007

Figure 16-42. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 17. Default messaging provider problem determination

Estimated time

00:45

What this unit is about

This unit describes how to troubleshoot Default Messaging in WebSphere Application Server V6.1.

What you should be able to do

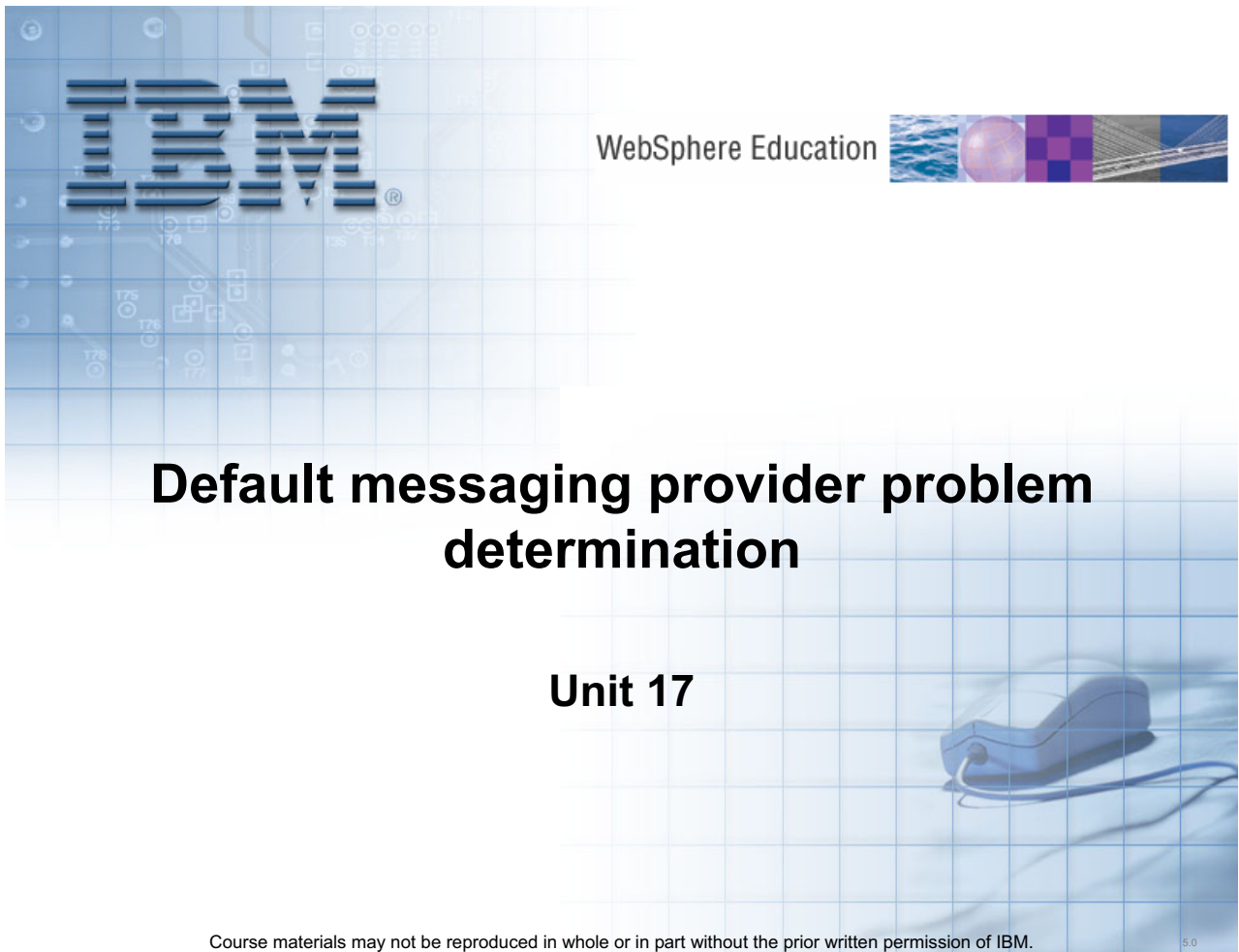
After completing this unit, you should be able to:

- Describe the components involved in default messaging
- Identify symptoms of default messaging related problems
- Collect diagnostic data
- Analyze diagnostic data and determine root causes
- Validate solutions

How you will check your progress

Accountability:

- Machine exercises



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.0

Figure 17-1. Default messaging provider problem determination

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit objectives

After completing this unit, you should be able to:

- Describe the components involved in default messaging
- Identify symptoms of default messaging related problems
- Collect diagnostic data
- Analyze diagnostic data and determine root causes
- Validate solutions

Figure 17-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

WebSphere Default Messaging

- WebSphere Application Server V6 delivered an all-new default implementation of JMS and an all new messaging engine called WebSphere Default Messaging.
 - WebSphere Default Messaging was formerly called by the following names:
 - Service Integration Bus or SIB
 - WebSphere Platform Messaging or WPM
 - Service integration technology or SIT
 - You may see references to the older names on the Web

- Fully J2EE 1.4 compliant
 - Supports JCA 1.5 and JMS 1.1 standards

- Provides a “Mediations” programming extension
 - Enables message modification and routing capabilities

Figure 17-3. WebSphere Default Messaging

WA5711.0

Notes:

WebSphere Default Messaging was introduced in WebSphere Application Server version 6. WebSphere Default Messaging includes a default JMS provider written purely in Java, allows both point to point and publish/subscribe messaging, and is fully integrated into the WebSphere Application Server. It was enhanced in version 6.1 to allow the use of a file system-based message store. The new file system-based message store is strongly recommended by IBM.

The J2EE 1.4 was developed under the Java Community Process. J2EE is defined under JSR 151.

Instructor notes:

Purpose — Introduce WebSphere Default Messaging.

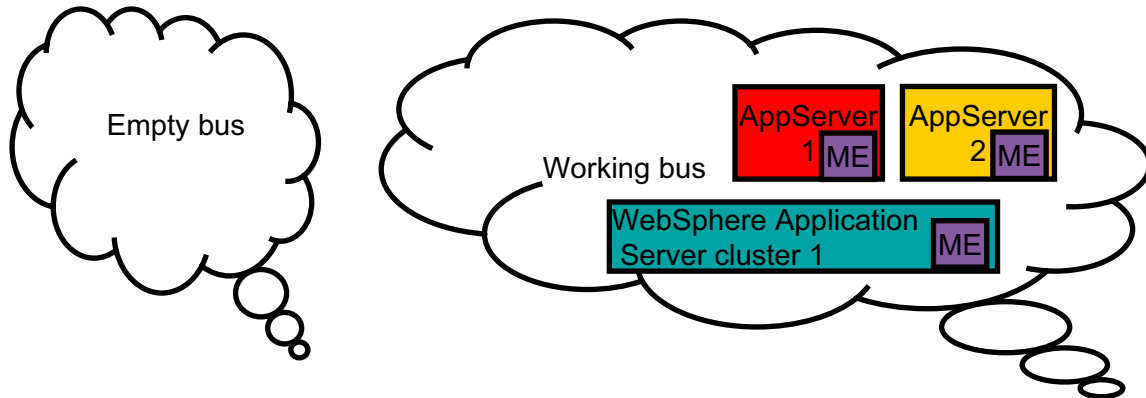
Details — It is sometimes referred to as SIB, which stands for System Integration Bus

Additional information — It replaces WebSphere MQ in the WebSphere Application Server world. WebSphere Application Server V5.1 was shipped with WebSphere MQ underneath that provided the messaging and JMS components.

Internal Note: Mention mediations to be complete as possible with just 45 minutes; however, remember the feature is not being used in the real world very much. It is thought to be redundant with such products as WebSphere Process Servers or WebSphere Enterprise Service Bus, among others, that do much more powerful mediation type work. It may or may not be there in version 7.

Transition statement — Concept view of a bus.

WebSphere Default Messaging concept view



- A bus is an administrative concept.
- It can do nothing at all until application servers and or clusters are added as members of the bus
- Once an application server or cluster is added to the bus as a member a messaging engine will be created.

Figure 17-4. WebSphere Default Messaging concept view

WA5711.0

Notes:

The messaging engine (ME), is what makes WebSphere Default Messaging work. The ME does the messaging work moving and storing messages.

Instructor notes:

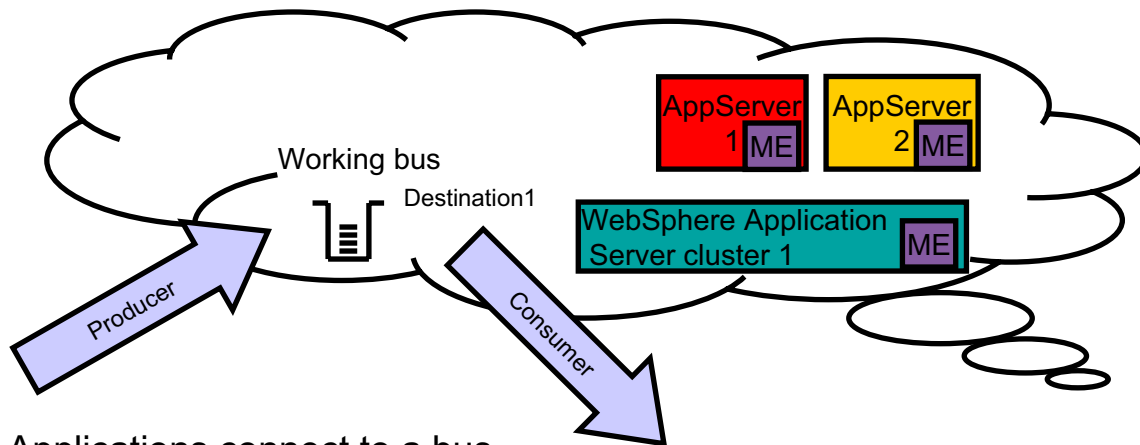
Purpose — Messaging concepts

Details — An ME is similar to a Queue Manager in WebSphere MQ. The two are similar, but not the same. The application server typically starts the ME, where as a user must start a Queue Manager. The ME is for the most part abstracted away from the user, where as in WebSphere MQ, the user must be intimately aware of the Q_Manager and the objects defined on it. ME-to-ME communications are handled internally by the ME, where as in WebSphere MQ, the user must define and manage connections between queue manager.

Additional information — WebSphere Default Messaging is not designed to replace WebSphere MQ. The two products are both messaging middleware, but they play in very different markets. In fact they work very well together.

Transition statement — The next slide, add a destination and some applications.

WebSphere Default Messaging concept view



- Applications connect to a bus.
- An application that sends messages attaches to a destination as a producer.
- An application that receives messages attaches to a destination as a consumer.
- A mediation is an application that is deployed to the bus, and associated with one or more destinations. It can be used for message manipulation and routing.

Figure 17-5. WebSphere Default Messaging concept view

WA5711.0

Notes:

The way to think about a bus is to imagine all of its members interconnected. This is contrast to the WebSphere MQ world, where a queue is in a static place, hosted by one, and only one Queue Manager. In WebSphere Default Messaging, the application connects to the bus as a whole and not an individual Queue Manager. As a result, all queues or topics in the bus are available to any application connected to the bus. It does not matter where the destinations were originally defined. Not pictured here is the concept of a message store. A message store is the place where a ME will store state information and persistent message data. In WebSphere Application Server Version 6 the message store can only be a database such as DB2 or Oracle. In WebSphere Application Server version 6.1, the message store can additionally be a stored on disk. If the message store is stored on disk, it is called a filestore. If it is stored in a database, it is called a data store. Using the filestore is recommended by IBM.

Instructor notes:

Purpose — This is the whole picture of a small bus in action.

Details — The blue arrows are representing application that are stand alone JMS clients consuming and producing messages to destination one. A key point in this picture is that the destination, kind of floats out in the bus. This is to show the separation between where the destination physically lives, and it overall availability. It does not matter if Destination one was defined on AppServer1, AppServer2, or Cluster one. As long as the cluster or server where the destination was defined is a member of the bus, then any application that connects to that bus can access it. This is a huge leap away from the WebSphere MQ world.

Additional information — This is a very simple bus. In the real world you might see customers using what is called a foreign bus. It was not covered to keep this lecture down to 45 minutes. A foreign bus is another bus that is connected to the local bus. A foreign bus can be either another WebSphere Default Messaging bus, or even a WebSphere MQ Q_Manager. In the case of a Q_Manager, the default messaging bus thinks the Q_Manager is another bus, and the Q_Manager thinks the bus is another Q_Manager.

Transition statement — Next a look at generic debug steps.

How to approach a generic problem with WebSphere Default Messaging

- Five steps for all messaging issues:
 - Check that the ME is in a started state.



- Check SystemOut.log for the specific exception you are trying to debug.
- Check the FFDCs for further details on the specific exception you are trying to debug.
- Check the configuration.
- Trace and engage IBM support.
- Specific problems
 - ME Startup problems
 - Message flow problems
 - Application connection problems

Figure 17-6. How to approach a generic problem with WebSphere Default Messaging

WA5711.0

Notes:

The next section of this lecture covers a general five step approach for all message problems. There will be a discussion in detail each of those steps. Then, some common problems in the specific area of ME Start-up, Message flow and application connections will be covered.

1. Check that the ME is started. If you have access to the administration panels, navigate to **Service Integration -> Buses -> BusName -> Messaging Engine** then look for the green arrow indicating the ME is running.
2. Check the SystemOut.log for the specific exception you are trying to debug.
3. Check the FFDCs for the specific exception you are trying to debug.
4. Check the configuration in the administration panels, or, if not available, check the XML configuration files.
5. Search the IBM Support pages, trace the problem and engage IBM Support.

Instructor notes:

Purpose — Map of the next section

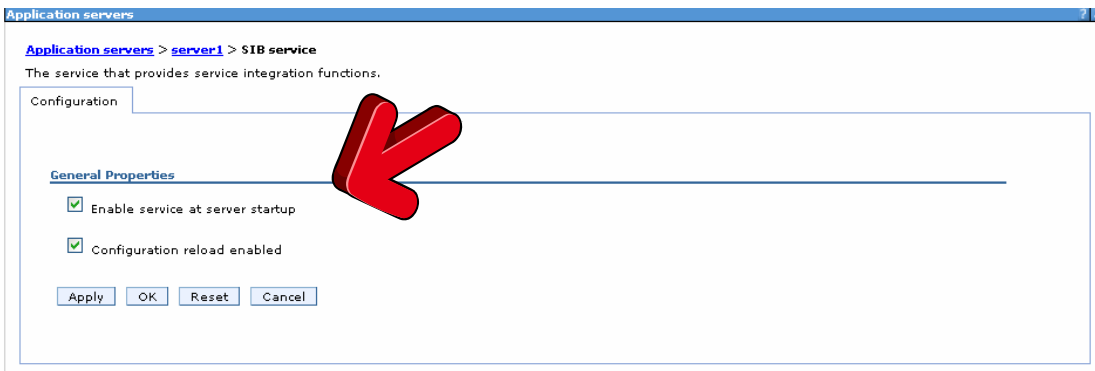
Details —

Additional information —

Transition statement — Step one

How to approach a generic problem with WebSphere Default Messaging – Step one (1 of 2)

- Check the state of the ME to be sure it started.
 - Search the SystemOut.log for the string 'SibMessage'.
- If a ME fails to start, WebSphere Default Messaging will not function.
 - If the SIB Service is disabled, the ME will not start and no messaging will occur.



- The next slide shows a ME starting successfully.

Figure 17-7. How to approach a generic problem with WebSphere Default Messaging – Step one (1 of 2)

WA5711.0

Notes:

Check that the ME has gotten to a state of **started**.

Make sure that the SIB service is enabled.

Instructor notes:

Purpose — Step One for all messaging problems is to make sure the ME is started.

Details —

Additional information — It is important to keep up with maintenance. If you are several fixpacks behind, it is good behavior to at least try the latest fix pack. Especially if you can recreate the issue in test.

Transition statement — More step one

How to approach a generic problem with WebSphere Default Messaging – Step one (2 of 2)

- The ME in this case goes from Joined to Starting.

```
SystemOut.log(3815): [4/28/07 11:12:46:298 EDT] 0000000f SibMessage      I
[ZMagicBus:Node03.server1-ZMagicBus] CWSID0016I: Messaging engine
Node03.server1-ZMagicBus is in state Joined.
```

```
SystemOut.log(3816): [4/28/07 11:12:46:819 EDT] 00000013 SibMessage      I
[ZMagicBus:Node03.server1-ZMagicBus] CWSID0016I: Messaging engine
Node03.server1-ZMagicBus is in state Starting.
```

- The ME then grabs its lock on its tables in the database.

```
SystemOut.log(3905): [4/28/07 11:12:59:136 EDT] 0000001c SibMessage      I
[ZMagicBus:Node03.server1-ZMagicBus] CWSIS1538I: The messaging engine,
ME_UUID=E9559146F69BE94B, INC_UUID=3114b5bc38c080f6, is attempting to
obtain an exclusive lock on the data store.
```

```
SystemOut.log(3906): [4/28/07 11:12:59:176 EDT] 0000001d SibMessage      I
[ZMagicBus:Node03.server1-ZMagicBus] CWSIS1543I: No previous owner was
found in the messaging engines data store.
```

```
SystemOut.log(3907): [4/28/07 11:12:59:227 EDT] 0000001c SibMessage      I
[ZMagicBus:Node03.server1-ZMagicBus] CWSIS1537I: The messaging engine,
ME_UUID=E9559146F69BE94B, INC_UUID=3114b5bc38c080f6, has acquired an
exclusive lock on the data store.
```

- The ME then goes into a state of Started and is ready for action.

```
SystemOut.log(3908): [4/28/07 11:13:04:704 EDT] 00000013 SibMessage      I
[ZMagicBus:Node03.server1-ZMagicBus] CWSID0016I: Messaging engine
Node03.server1-ZMagicBus is in state Started.
```

Figure 17-8. How to approach a generic problem with WebSphere Default Messaging – Step one (2 of 2)

WA5711.0

Notes:

A healthy ME will complete this cycle upon each startup of the application server that hosts it. This is the first thing to check when debugging any WebSphere Default Messaging problem. Search the SystemOut.log for the string SibMessage.

Instructor notes:**Purpose —****Details —**

The ME issues a start by the Application Server.

The ME will go into a state of Joined. This means it simply has made contact with the workload manager.

The ME will go into a state of starting, meaning it now is beginning to gather the needed resources it needs to be considered started.

The ME will then grab a lock on the database tables it is using as a message store. This step is a bit different if you are using filestore instead of data store. If using data store, then a lock must be gained in order for the ME to go further along in starting. If some other ME has the lock, or if the tables do not exist, the ME will retry for a period of time and then give up, putting itself in a state of 'disabled'. Manual intervention will be needed at this stage.

The ME has now gone into a state of started and will begin doing its work.

Additional information — In a cluster you will see an ME get to a joined state and not go further. This means that same ME is running on some other server in the cluster. When searching the SystemOut, search on the string 'SibMessage'.

Transition statement — Step two

How to approach a generic problem with WebSphere Default Messaging – Step two

- Use your favorite tool to search the SystemOut.log for the exception you are trying to debug.
- Read the details of the exception and any stack trace that might be associated with the exception
- Look at the package name to determine what component is involved in the failure.
 - This will give you an idea on what to do next.

```
[9/3/07 10:20:15:271 EDT] 00000012 XAREcoveryData WTRN0005W: The XAResource for a transaction participant could not be recreated and transaction recovery may not be able to complete properly. The resource was com.ibm.ws.sib.processor.impl.store.MsgStoreXAResourceInfo@653c653c. The exception stack trace follows: com.ibm.ws.Transaction.XAResourceNotAvailableException: com.ibm.websphere.sib.exception.SIResourceException: CWSIT0019E: No suitable messaging engine is available on bus Avalon that matched the specified connection properties {busName=Avalon, targetType=MEUuuid, targetGroup=8EEEEAA28785D48FD, targetSignificance=Required}. Reason for failure: CWSIT0103E: No messaging engine was found that matched the following parameters: bus=Avalon, targetGroup=8EEEEAA28785D48FD, targetType=MEUuuid, targetSignificance=Required, transportChain=InboundBasicMessaging, proximity=Bus.
    at
    com.ibm.ws.sib.processor.impl.store.xarecovery.MsgStoreXAResourceFactory.getXAResource(MsgStoreXAResourceFactory.java:117)
    at com.ibm.ws.Transaction.JTA.XAREcoveryData.getXARminst(XAREcoveryData.java:529)
    at com.ibm.ws.Transaction.JTA.XAREcoveryData.recover(XAREcoveryData.java:644)
    at com.ibm.ws.Transaction.JTA.PartnerLogTable.recover(PartnerLogTable.java:524)
    at com.ibm.ws.Transaction.JTA.RecoveryManager.resync(RecoveryManager.java:1830)
    at com.ibm.ws.Transaction.JTA.RecoveryManager.run(RecoveryManager.java:2509)
    at java.lang.Thread.run(Thread.java:797)
```

Figure 17-9. How to approach a generic problem with WebSphere Default Messaging – Step two

WA5711.0

Notes:

In the example stack trace you can see that the component involved in this error is blank. Since you now know the error is in the component processor, then you know you are dealing with a problem in the core processing classes. Clearly there is a problem in the core messaging classes; messages are not flowing properly and you need to know why. This exception tells the tale and is basic as “I just shut down the ME for this bus and it is complaining about CWSIT0019E: No suitable messaging engine is available.”

If the exception was being reported from say sib.JMSRa then you would know the problem is with the JMS resource adaptor. A problem with the sibJMSRa component would indicate a problem with an client application or MDB talking to the bus.

Instructor notes:

Purpose — Clues to determining the component involved in the problem by looking at an exception in the SystemOut.log.

Details — Later in this lecture there is a discussion about the various components that make up the bus code.

Additional information —

Transition statement — Step three

How to approach a generic problem with WebSphere Default Messaging – Step three

- FFDCs report further details of an exception reported in the SystemOut.
- If installed using the defaults, on Windows you will find the FFDCs in the following directory.
 - C:\Program Files\IBM\WebSphere\AppServer61\profiles\ProfileName\logs\ffdc

```
Stack Dump = com.ibm.ws.Transaction.XAResourceNotAvailableException: com.ibm.websphere.sib.exception.SIResourceException: CWSIT0088E:
There are currently no messaging engines in bus Avalon running. Additional failure information: CWSIT0103E: No messaging engine was found
that matched the following parameters: bus=Avalon, targetGroup=DB59DAB34C3A9BD8, targetType=MEUUid, targetSignificance=Required, transport
  at com.ibm.ws.sib.processor.impl.store.xarecovery.MsgStoreXAResourceFactory.getXAResource(MsgStoreXAResourceFactory.java:117)
  at com.ibm.ws.Transaction.JTA.XARecoveryData.getXARminst(XARecoveryData.java:529)
  at com.ibm.ws.Transaction.JTA.XARecoveryData.recover(XARecoveryData.java:644)
  at com.ibm.ws.Transaction.JTA.PartnerLogTable.recover(PartnerLogTable.java:524)
  at com.ibm.ws.Transaction.JTA.RecoveryManager.resync(RecoveryManager.java:1830)
  at com.ibm.ws.Transaction.JTA.RecoveryManager.run(RecoveryManager.java:2509)
  at java.lang.Thread.run(Thread.java:797)
Caused by: com.ibm.websphere.sib.exception.SIResourceException: CWSIT0088E: There are currently no messaging engines in bus Avalon running
Additional failure information: CWSIT0103E: No messaging engine was found that matched the following parameters: bus=Avalon,
targetGroup=DB59DAB34C3A9BD8, targetType=MEUUid, targetSignificance=Required, transportChain=InboundBasicMessaging, proximity=Bus.
  at com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.localBootstrap(TrmSICoreConnectionFactoryImpl.java:351)
  at com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.createConnection(TrmSICoreConnectionFactoryImpl.java:292)
  at com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.createConnection(TrmSICoreConnectionFactoryImpl.java:160)
  at com.ibm.ws.sib.processor.impl.store.xarecovery.MsgStoreXAResourceFactory.getXAResource(MsgStoreXAResourceFactory.java:105)
```

Figure 17-10. How to approach a generic problem with WebSphere Default Messaging – Step three

WA5711.0

Notes:

WebSphere Application Server cuts many FFDCs. Most will not be relevant to the problem that is being debugged. The best way to sort through the many FFDCs that are cut is to match up the timestamp with the exception report in the SystemOut.log. Another approach is to search the FFDC directory for the specific exception or even component name.

Instructor notes:

Purpose — How to use FFDCs to help in messaging cases.

Details — Typically, it is most helpful to look at the stack. In this example, there is a `SIResourceException` complaining about the fact that there is no messaging engine in the bus Avalon. There is no messaging engine in Avalon, because all of the bus members were deleted. To solve this problem, you just need to add a member to the bus and a new ME will be created.

Additional information —

Transition statement — Step four

How to approach a generic problem with WebSphere Default Messaging – Step four

- Sib-destinations.xml
 - This is the file to look at for destinations and their localization points.
 - Found in C:\Program Files\IBM\WebSphere\InstallDir\profiles\ProfileName\config\cells\CellName\buses\BusName
- Sib-bus.xml
 - This file lists first the name of the bus, then a list of bus members, followed by any foreign buses and their link definitions.
 - Found in C:\Program Files\IBM\WebSphere\InstallDir\profiles\ProfileName\config\cells\CellName\buses\BusName
- Sib-engines.xml
 - This file is has information regarding the ME, the datastore, the localization points, and the gateway links.
 - Found in C:\Program Files\IBM\WebSphere\installDir\profiles\ProfileName\config\cells\CellName\nodes\NodeName\servers\ServerName

Figure 17-11. How to approach a generic problem with WebSphere Default Messaging – Step four

WA5711.0

Notes:

In cases where the system has never worked and is being set up for the first time, check the configuration. If you have access to the administration console check the configuration by looking at the administration panels. If you do not have access to the administration console, then you will have to look at the configuration files.

Look at the Sib-destinations file if an application is failing to produce or consume messages from a destination. Check to make sure that the destination is actually there and double check the spelling, including case. Take a look at the Sib-bus file for a sanity check. Check to see that the bus is created and the bus members you defined exist. The Sib-engines file shows the most information in one place. In Sib-engines, you can check the messaging engine definition and its message store information. This is very useful if you are having trouble getting an ME to start. The Sib-engines file also lists the localization points for queues and topics localized to each specific ME, and any foreign bus links.

Instructor notes:

Purpose — Familiarize students with the most important XML files that are created and used by WebSphere Default Messaging.

Details — If the problem is destination not found on produce or consume look at the destinations file for spelling and case.

If the problem is a failure to connect to a bus, check the Sib-bus file to make sure the bus exists and check spelling and case.

If the problem is a ME will not start, check the Sib-engines file to check on the data store definition.

If the problem is related to producing and consuming messages in a foreign bus, or even just connecting to a foreign bus, check the Sib-engines file.

Additional information — “Found IN” is where these would be if you use the defaults and install on Windows. It is worth mentioning to students that all of the configurations checks can be done in the administrative panels. Support does not have access to them, so you are forced to use the XML files. If the student does not have access to the administrative panels for a particular case, they will be forced to use the XML files.

Transition statement — More step four

How to approach a generic problem with WebSphere Default Messaging – Step four

• Sib-engines.xml Sample

```
<?xml version="1.0" encoding="UTF-8" ?>
<sibresources:SIBMessagingEngine xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:sibresources="http://www.ibm.com/websphere/appserver/schemas/6.0/sib
resources.xmi" xmi:id="SIBMessagingEngine_1183049318078"
  name="legendNode01.server1-Avalon" uuid="D23A534EB3112CEA"
  busName="Avalon" busUuid="39F7AE063953A64C" highMessageThreshold="50000">
  <dataStore xmi:id="SIBDatastore_1183049318088" uuid="75F42F88DAD1E31D"
  dataSourceName="jdbc/com.ibm.ws.sib/legendNode01.server1-Avalon" />
  <localizationPoints xmi:type="sibresources:SIBQueueLocalizationPoint"
  xmi:id="SIBQueueLocalizationPoint_1183049319179"
  identifier="_SYSTEM.Exception.Destination.legendNode01.server1-
Avalon@legendNode01.server1-Avalon" uuid="A52A703F3BBD84DC"
  targetUuid="4F2475E165050BFD0DDA3454" />
  <localizationPoints xmi:type="sibresources:SIBTopicSpaceLocalizationPoint"
  xmi:id="SIBTopicSpaceLocalizationPoint_1183049319199"
  identifier="Default.Topic.Space@legendNode01.server1-Avalon"
  uuid="B0570E08A3A67965" targetUuid="40E26753050ED698557E3273"
  highMessageThreshold="50000" />
  <localizationPoints xmi:type="sibresources:SIBQueueLocalizationPoint"
  xmi:id="SIBQueueLocalizationPoint_1184336421686"
  identifier="Excaliber@legendNode01.server1-Avalon" uuid="26EC6E658045CF68"
  targetUuid="D89E7068A5AB17FB382F3D13" highMessageThreshold="50000" />
</sibresources:SIBMessagingEngine>
```

Figure 17-12. How to approach a generic problem with WebSphere Default Messaging – Step four

WA5711.0

Notes:

In this example, the Sib-engines file shows that there is an ME called **legendNode01.server1-Avalon** with a **UUID** of **D23A534EB3112CEA**. In this example, the Sib-engines file shows that the MEs data store has an ID of **75F42F88DAD1E31D** and that it has a **dataSourceName** of **jdbc/com.ibm.ws.sib/legendNode01.server1-Avalon**.

In this example, the Sib-engines file shows that there are three localization points. The first on the list is the

SYSTEM.Exception.Destination.legendNode01.server1-Avalon@legendNode01.server1-Avalon. This is the default exception destination for undeliverable messages. The second is the **Default.Topic.Space@legendNode01.server1-Avalon**. This is the default topic space and just like the system exception destination it is defined by the ME at creation time. The third and final localization point is a user defined queue called **Excaliber@legendNode01.server1-Avalon**. The queue itself is actually named Excalibur. The localization point name adds its physical location in the format of **NodeName.ServerName-BusName**.

Instructor notes:

Purpose — Take a quick look at a simple Sib-engines file

Details — A UUID is an unique user Identification. It is the internal name for the object. It can be used in trace and in the SystemOut.log to search for a specific object.

Additional information — It takes time and experience to get something out of these XML files. Create objects one at a time and look to see what gets added to the XML files to get familiar with what goes where.

Transition statement — Step five

How to approach a generic problem with WebSphere Default Messaging – Step five

- Search the IBM support site for clues to a solution for the problem.
 - <http://www-306.ibm.com/software/webservers/appserv/was/support/>
- If you have gotten to this point and have not found a solution it is time to trace the problem. Trace should always be your last resort.
- Reading default messaging trace is very difficult. It is near impossible without access to the code. For this reason it is best to consult IBM support once the problem requires trace to solve.
- To get the best support from IBM, one should prepare a clear problem statement and collect the appropriate documentation before you open a PMR.
 - <http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg21145599>

Figure 17-13. How to approach a generic problem with WebSphere Default Messaging – Step five

WA5711.0

Notes:

WebSphere Default Messaging Technotes are in the same place as all other WebSphere Application Server Technotes.

<http://www-306.ibm.com/software/webservers/appserv/was/support/>

Reading trace is not easy, and without access to the source code, it is even more difficult. For this reason you will likely need to consult IBM Support.

Before you call IBM Support prepare the following:

1. Full problem description with details on how the problem happened including what changed just before the problem occurred.
2. At the very least have the SystemOut.log ready for submission.
3. It will speed resolution time if you have a collector tool output, and even trace, ready for submission.

Optional: If you can recreate the problem, send in a test case that produces the problem with recreation steps.

Instructor notes:

Purpose — Opening a PMR with IBM

Details — Optional:

If you can recreate the problem, send in a test case that produces the problem with recreation steps.

If confidentiality is a problem the customer should send in just the SystemOut, FFDCs, SystemErr, and any XML files they are willing to part with.

Additional information — The biggest key to getting good support in any situation is complete and relevant information. It is no different with IBM Support. The more complete and relevant the information you provide on your problem the better support you will receive.

Transition statement — ME start up problems

How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (1 of 4)

- It will not always be apparent at application server start-up that the ME did not start.
- If the messaging engine is stopped, check the log for the following messages:
 - Message CWSID00016I
 - This message tells you that the ME is stopped. Other messages will likely be present to help determine why the ME is stopped.
 - Message CWSIS0002E
 - This message might contain the information that tells you the cause of the problem, or you might have to scan back through the log to find more information.
 - Message CWSID0027I
 - This message indicates that a serious problem occurred during ME start-up or restart.

Figure 17-14. How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (1 of 4) WA5711.0

Notes:

Search the SystemOut.log for SibMessage. This will show the sequence of the states the ME got into on its way to starting. Once it is known what state the ME got to, look at the message to see why it did not move further along.

Instructor notes:

Purpose — Debugging ME startup hints.

Details — Trace rarely helps in a ME startup problem. Usually ME startup problems are configuration issues. The XML files are useful in these cases.

Additional information — Support will ask for the collector tool output.

Transition statement — Common configuration ME startup failures.

How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (2 of 4)

- Common causes of ME startup failures.
 - JNDI name error when accessing the messaging data store.
 - Invalid authentication alias information.

The screenshot shows the 'Configuration' console with the 'General Properties' tab selected. The 'Data source JNDI name' field is highlighted in yellow and has a red arrow pointing to it. The 'Authentication alias' dropdown menu is also highlighted with a red arrow. Other fields include UUID (75F42F88DAD1E31D), Schema name (IBMWSSIB), and checkboxes for 'Create tables'. The 'Number of tables for permanent objects' and 'Number of tables for temporary objects' are both set to 1. The 'Apply', 'OK', 'Reset', and 'Cancel' buttons are at the bottom.

Figure 17-15. How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (2 of 4) WA5711.0

Notes:

A JNDI naming problem is usually caused by an incorrect value in the data source JNDI name field of the data store definition. The solution is to correct the data source JNDI name field.

An invalid authentication alias problem is usually caused by an incorrect component managed authentication alias. This alias can be specified for the data source definition, or for the messaging engine definition. The solution is to correct the alias.

Navigate to: **Buses -> BusName -> Messaging engines -> MENAME -> Data store**

Instructor notes:

Purpose — Common configuration issues that will cause a ME to not start.

Details — The arrows in the picture show where to check the data source JNDI name and the authentication alias.

Additional information —

Transition statement —

How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (3 of 4)

- Common causes of ME startup failures.
 - Invalid database tables or tables with invalid data.
 - Each messaging object has a unique identifier associated with it that is called a UUID. When a messaging engine is created and first accesses the database, the messaging engine registers its UUID in the database SIBOWNER table. When a database has been marked in such a way, it cannot be used by another messaging engine.
 - The solution is to drop and recreate the database tables.

Configuration

General Properties

UUID
75F42F88DAD1E91D

Data source JNDI name
jdbc/com.ibm.ws.sib/legendNode01.server1-Avalon

Schema name
IBMWSSIB

Authentication alias
(none)

Create tables

Number of tables for permanent objects
1

Number of tables for temporary objects
1

Apply OK Reset Cancel

Related Items
JAAS - J2C authentication data

Figure 17-16. How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (3 of 4) WA5711.0

Notes:

The most common reasons for having an incorrect UUID in the data store tables are an administrator deletes an application server or removes it from the bus, and then tries to use the same database when adding a new server to the bus, and an administrator creates a bus, deletes it, and then creates a new bus with the same name and configuration.

Instructor notes:

Purpose — Common caused of an ME not getting a lock on its tables in the database.

Details — The picture shows where you can give the ME permission to create tables in the database if it is discovered they are not there. The database administrator will have to grant create tables permission to the ME. Most database administrators are reluctant to grant this powerful a permission to anyone. It is likely that you will have to ask the database administrator to recreate the tables for you if the need arises.

Additional information — At 6.1 IBM recommends filestore, so this is not an issue. However, most in the field right now are still using a database as a message store.

Transition statement —

How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (4 of 4)

- Common causes of ME startup failures.
 - Message store not available.
 - If an ME can not contact the data store, then the ME will not start.
 - Ensure that the database is started correctly.
 - Attempt to restart the messaging engine.
 - If the messaging engine will not start, stop and restart the application server.
 - Test connectivity to the messaging data store database.
 - > Note that this will not work if the database is secured and the authentication alias was defined at the messaging engine data store level.

Figure 17-17. How to approach a specific problem with WebSphere Default Messaging – ME start-up problems (4 of 4) WA5711.0

Notes:

Message store not available means the ME called and the message store did not answer. If the message store is remote, then it could be a TCP/IP issue.

Instructor notes:

Purpose — More ME start up problems

Details — FFDCs tend to help in these cases. They will provide more details on why the generic message store not available exception.

Additional information —

Transition statement — Message flow problems

How to approach a problem with WebSphere Default Messaging – Message flow problems (1 of 3)

- Common problems that cause message flow issues.
 - The message was produced, but now it cannot be found
 - WebSphere Default Messaging uses internal queues to move messages between applications and MEs and between MEs.
 - It is possible that your messages is on one of these internal queues and you can not see it.
 - There are several recent defects in this area so first check your fix pack level.
 - Try a restart of the application server.

- Common problems that can cause message flow issues.
 - Producing messages works fine, but the consumer cannot get them.
 - It is possible that the ME where the queue point of the destination is hosted but is not running.
 - The solution is to check to make sure the ME that is hosting the queue point is running.

Figure 17-18. How to approach a problem with WebSphere Default Messaging – Message flow problems (1 of 3)

WA5711.0

Notes:

There are many possible scenarios where message flow is a problem. One could argue that all problems in messaging are message flow problems because, for example, if a ME does not start messages will not flow. A queue point is the administrative view of a localization. All queues and topics have a queue point or localization. The queue point is where the object is physically located.

Instructor notes:

Purpose — Message flow issues

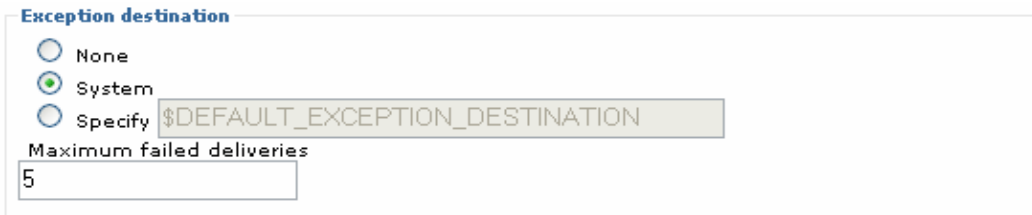
Details —

Additional information — A destination is an administrative concept. The messages on the destination are physically located in what is called a queue point. On WebSphere Application Server V6.0 the queue point lives in a database, and in V6.1 it is located in either in a database, or a filestore.

Transition statement — More message flow problems.

How to approach a problem with WebSphere Default Messaging – Message flow problems (2 of 3)

- Common problems that can cause message flow issues.
 - Producing messages works fine, but the consumer can not get them.
 - Check the exception destination because messages that failed to be delivered may have been put there.



The screenshot shows the 'Exception destination' configuration in WebSphere. It features three radio buttons: 'None', 'System', and 'Specify'. The 'System' radio button is selected. To the right of the 'Specify' radio button is a text field containing the value '\$DEFAULT_EXCEPTION_DESTINATION'. Below these options is a label 'Maximum failed deliveries' followed by a text input field containing the number '5'.

- Messages appear to be getting lost.
 - WebSphere Default Messaging will lose messages by design depending on quality of service.
- In general, the higher the quality of service the lower the performance.

Figure 17-19. How to approach a problem with WebSphere Default Messaging – Message flow problems (2 of 3)

WA5711.0

Notes:

If the MDB fails to handle the message being delivered to it, the ME will attempt to redeliver the message. The number of times the ME will attempt to redeliver the message to the MDB is driven by maximum failed deliveries parameter of the destination. Once the maximum number of retries is exhausted, the message will be delivered to the defined exception destination. If there is no exception destination defined. The maximum failed deliveries parameter will be ignored resulting in an infinite loop when the message is not deliverable to the MDB.

Instructor notes:

Purpose — More message flow issues

Details —

Additional information — The picture shows a part of a destination definition. It shows the part where you can set maximum retries and what exception destination to use.

Transition statement — Quality of service

How to approach a problem with WebSphere Default Messaging – Message flow problems (3 of 3)

- The following table shows the various quality of service levels of WebSphere Default Messaging.
- If you think you are losing messages, check the quality of service setting, because it may be working as designed.

	ME Failure	ME Shutdown	Comms Failure	High Load
Best effort non-persistent	Discarded	Discarded	May be discarded	May be discarded
Express non-persistent	Discarded	Discarded	May be discarded	Preserved
Reliable non-persistent	Discarded	Discarded	Preserved	Preserved
Reliable Persistent	May be discarded	Preserved	Preserved	Preserved
Assured Persistent	Preserved	Preserved	Preserved	Preserved

Figure 17-20. How to approach a problem with WebSphere Default Messaging – Message flow problems (3 of 3)

WA5711.0

Notes:

On the chart above, **Discarded** means the message will be discarded, **May be discarded** means the message may be discarded, and **Preserved** means the message will not be discarded.

Instructor notes:

Purpose — Quality of service

Details — Best effort non-persistent messages will never hit the message store. Use this setting if you want ultra fast messages that are not important.

Express non-persistent messages may be sent to the message store under high load. Use this setting if you need speed but also you want some measure of reliability.

Reliable non-persistent messages are saved in the message store in all cases except when the ME stops, either from a planed shutdown or crash. This is the most reliable of the non-persistent message options.

Reliable persistent messages will be saved in the message store in almost all cases. There is a small chance you could lose a reliable persistent message in the case of a crash of the ME.

Assured persistent message will never be lost. This is also the slowest of the quality of service options because all messages hit the message store which required disk writes.

Additional information —

Transition statement — Application connection issues

How to approach a problem with WebSphere Default Messaging – Application connection issues

- Common causes for application connection issues.
 - A client application receives a report of CWSIT0006E.
 - This indicates that the bus the application is trying to connect to is not available.
 - A client application receives a ServiceUnavailableException.
 - Check to make sure the application server the application is connecting to is started.
 - Check the bootstrap address and port number for accuracy.

Notes:

In the case of a report of CWSIT0006E, check the bus name in the connection factory definition for the case sensitive spelling of the bus name, check the scope of the connection factory, and check to make sure there is an ME running on the bus.

Instructor notes:

Purpose — Application connection issues

Details — The green arrow on the ME indicated that it is in a started state and ready for action. Scope is important. If the object is defined at too small of a scope then things will not work properly. This is a common mistake with new users of WebSphere Default Messaging. The general rule is to use cell scope unless you have a very specific need to be more granular in regards to scope.

Additional information — This will be the subject of the exercise.

Transition statement — Components and trace groups

Components and their trace groups (1 of 3)

- Sib.jms
 - This is the WebSphere Default Messaging JMS provider code.
 - The trace group for Sib.jms is **SIBJms*=all**.
- Sib.comms
 - This is the communications code in WebSphere Default Messaging. It handles both client to ME, and ME to ME communication.
 - Trace string is **SIBCommunications=all**.
- Sib.admin
 - This component provides the WebSphere Default Messaging administrative objects, for example buses, destinations and so on, which fit into the WebSphere Application Server administrative framework.
 - Trace group is **SIBAdmin*=all**.

Figure 17-22. Components and their trace groups (1 of 3)

WA5711.0

Notes:

- Sib.jms = Service Integration Bus dot Java Messaging Service
- Sib.comms = Service Integration Bus dot communications
- Sib.admin = Service Integration Bus dot administration

Instructor notes:

Purpose — Component recognition

Details — This section is to expose the student to the shortened package names of the components involved in default messaging. The student will then be able to tell when looking at the SystemOut.log which part of default messaging they are looking at and what it is responsible for doing.

Additional information —

Transition statement — More components

Components and their trace groups (2 of 3)

- Sib.mfp
 - This is the part of WebSphere Default Messaging that does message formatting and parsing. It will change the format of a message based on what the receiver expects. A good example of when this code is active is when the bus is talking to a WebSphere MQ Q_Manager.
 - Trace for this is **SIBMfp*=all**.
- Sib.messagestore
 - This is the part of WebSphere Default Messaging that does the SQL calls to the database. It internally does all of the XA work and does not use XA when speaking to the database.
 - Trace for this is **SIBMessageStore=all**.
- Sib.processor
 - This is the code that defines the ME. It is the “heart and soul” of WebSphere Default Messaging.
 - The trace for this is **SIBProcessor=all**.
- Sib.security
 - This is the security piece of WebSphere Default Messaging. The WebSphere Application Server OAM does all of the work in the area of security, so this bit just interfaces with the WebSphere Application Server OAM.
 - Trace for this is **SIBSecurity=all**.

Figure 17-23. Components and their trace groups (2 of 3)

WA5711.0

Notes:

- Sib.mfp = Service Integration Bus dot Message Formatting and Parsing
- Sib.messagestore = Service Integration Bus dot Message Store
- Sib.processor = Service Integration Bus dot Processor
- Sib.security = Service Integration Bus dot Security

Instructor notes:

Purpose — More components

Details —

Additional information —

Transition statement — More components

Components and their trace groups (3 of 3)

- Sib.mediations
 - This code does the mediation handler work.
 - The trace for this is **SIBMediations=all**.
- Sib.psb
 - This is the pub/sub bridge. It handles publish/subscribe messaging over MQLink.
 - The trace group is **SIBPsb=all**.
- Sib.trm
 - The Sib.trm is the topology routing manager. It relies heavily on the WebSphere application component called the workload manager (WLM). The WLM is responsible for deciding what MEs run where. The WLM then tells the SIBTrm what to do. The WLM also makes the decision as to what shared queue gets a message in a work load balancing cluster environment. It is common in Sib.trm problems to need the WebSphere Application Server WLM trace as well. In these cases, WebSphere Application Server level 2 support will be involved as well.
 - **WLM*=all** and **SIBTrm=all**.
- Sib.webservices
 - This component does all the Web service tasks. There is no trace group for this component. You will need to specify a trace strings. Consult IBM support on a case by case basis for the proper trace string. At the very least you will need to trace **com.ibm.ws.sib.webservices.***.

Figure 17-24. Components and their trace groups (3 of 3)

WA5711.0

Notes:

- Sib.mediations = Service Integration Bus dot Mediations
- Sib.jms = Service Integration Bus dot Publish Subscribe Bridge
- Sib.trm = Service Integration Bus dot Topology Routing Manager
- Sib.webservices = Service Integration Bus dot WebServices

Instructor notes:

Purpose — More components

Details —

Additional information — Does everyone know the difference between a trace group in WebSphere Application Server and a trace string? If not, you can say that a trace groups is a variable in the code that will turn on tracing for several packages on one go. A trace string will turn on trace just for that particular package, or portion of package.

Transition statement — SIB*=all warning

Tracing for WebSphere Default Messaging (1 of 3)

- If you trace SIB*=all you trace out every component, every class, and every method.
- If you do trace SIB*=all expect that it could take some time for support to read through, it will effect your systems performance, and it will produce mountains of data.
- Only use SIB*=all under the tutelage of IBM Support.

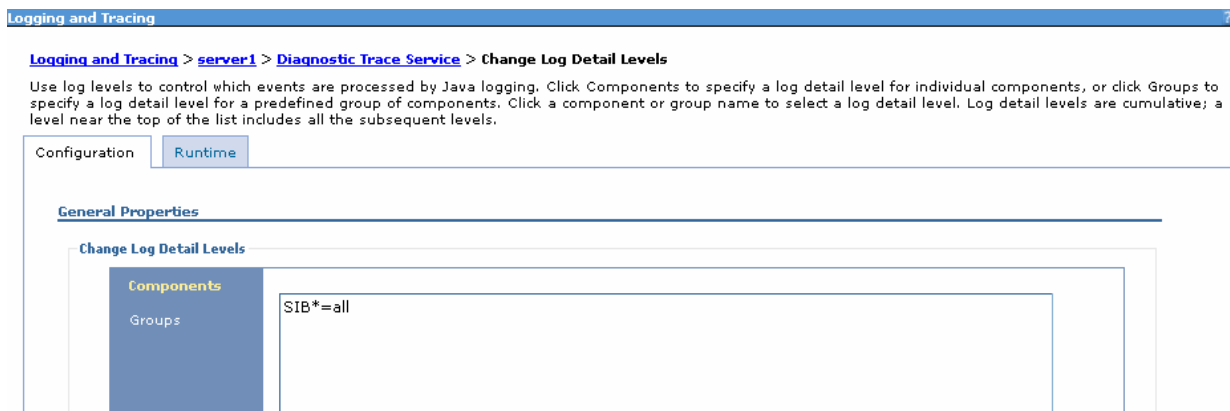


Figure 17-25. Tracing for WebSphere Default Messaging (1 of 3)

WA5711.0

Notes:

If you turn on SIB*=all tracing you will need to increase the number of historic files as this trace is super-verbose. You should expect a performance hit.

Instructor notes:

Purpose — Trace notes

Details — The screen capture shows where to set the level of tracing you want. This should be familiar to most students.

Additional information — WebSphere Default Messaging is a very big piece of code. In terms of lines of code it is nearly the size of all of WebSphere Application Server Version 5.

Transition statement — Heavy duty trace groups

Tracing for WebSphere Default Messaging (2 of 3)

- The following trace groups are the most verbose. Combining several of these trace groups may impact performance.
 - SIBMessageStore=all
 - This is the code that does the interaction with the database. It does the actual SQL calls to the DB.
 - SIBProcessor=all
 - This is the ME doing messaging. This is the engine of WebSphere Default Messaging.
 - SIBCommunications=all
 - This is the communications piece. This code handles ME to ME and Application to ME communication over TCP/IP.
 - SIBMfp=all
 - This does the message formatting and parsing.

Figure 17-26. Tracing for WebSphere Default Messaging (2 of 3)

WA5711.0

Notes:

Using two or more of the listed trace groups may effect performance in a negative way, sometimes to the point of not allowing the problem you are trying to trace to happen.

Instructor notes:

Purpose — Warn students of the extra heavy trace groups.

Details — Ask IBM Support to advise when using two or more of these trace groups.

Additional information —

Transition statement — What trace groups to use when?

Tracing for WebSphere Default Messaging (3 of 3)

- How do you know what trace groups to use?
 - You can use the stack trace from the SystemOut to decide what packages to trace.
 - Consult the Must Gather pages at the following link.
 - Go to the IBM WebSphere support pages and look at the MustGather section.
 - Look at the next few slides of this module.
 - When in doubt consult IBM Support.

Notes:

Looking at the stack trace of an exception will tell you what components are involved in the area where the problem occurred. Match up the components in the stack trace with the appropriate trace groups and there is a very good chance you will capture the cause of the problem in trace.

Instructor notes:

Purpose — Tips on how to determine what trace group to use for a specific problem.

Details —

Additional information —

Transition statement — The rest of the slides in this lecture list specific categories of problems and the suggested trace groups to capture them.

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (1 of 6)

- Problem – Where is the message? Or why is it so slow? Or why is it stuck on a transmission queue or the exception destination?
 - SIBCommunications=all;SIBMessageTrace=all;SIBProcessor=all
 - If there are MDBs involved (and it seems there often are), also use SIBRa*=all;SIBJmsRa*=all
 - You will also need the WebSphere MQ traces if WebSphere MQ is involved through the MQLink.

- Problem – JMS client problem to the bus.
 - SIBJms*=all:SIBJFapChannel=all:SIBCommunications=all:SIBMessageTrace=all:SIBTrm=all
 - You will need to trace both the server and the client.
 - Client trace instructions
- http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.wsfep.multiplatform.doc/info/ae/ae/ttrb_entrstandal.html

Figure 17-28. Tracing for WebSphere Default Messaging – What to trace for specific types of problems (1 of 6)

WA5711.0

Notes:

Where is your message? Or why is it so slow? Or why is it stuck on a transmission queue or the exception destination? It is useful to have the UUID of the message in question, especially if you are going to contact IBM Support.

It is normally not necessary to trace JMS client problems to the bus. Using the systemOut.log and checking the settings in the XML files will typically solve these types of issues.

Instructor notes:

Purpose — This section is to act as a guide for students to know what groups to use in what situation.

Details —

Additional information —

Transition statement — More what groups to use when.

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (2 of 6)

- Problem – There is a problem with messages, or MEs talking to the database, or message store.
 - SIBMessageStore=all:SIBProcessor=all
 - There is a chance you may need to produce a dump of the ME for message store problems. IBM support will direct you should you need to dump the ME.

- Problem – There is a problem in the ME. It is not delivering messages properly or just behaving badly in some manner.
 - SIBMessageTrace=all:SIBProcessor=all
 - The processor code is big. It is arguable the most important code in WebSphere Default Messaging. If there is a problem in this code, you will need to consult IBM support.

Notes:

Instructor notes:

Purpose — This section is to act as a guide for students to know what groups to use in what situation.

Details —

Additional information —

Transition statement — More what groups to use when.

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (3 of 6)

- Problem – There is a problem with security. A user ID or password is not working or the problem only happens once security is turned on.
 - SIBSecurity=all
 - This will trace out both authentication and authorization trace.
- Problem – There is a problem using Web services and it is not working as expected.
 - There are no Web services trace groups, so you will need to use trace strings.
 - com.ibm.ws.sib.webservices.*=all:com.ibm.ws.webservices.wssecurity.*=all:com.ibm.xml.soapsec.*=all:com.ibm.wsspi.wssecurity.*=all:com.ibm.ws.sibws.*=all

Figure 17-30. Tracing for WebSphere Default Messaging – What to trace for specific types of problems (3 of 6)

WA5711.0

Notes:

Instructor notes:

Purpose — This section is to act as a guide for students to know what groups to use in what situation.

Details —

Additional information —

Transition statement — More what groups to use when.

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (4 of 6)

- Problem – There is a problem putting or getting messages from the bus to a MQ Q_Manager, or from the MQ Q_Manager to the bus.
 - SIBCommunications=all:SIBMfp=all:SIBMqFapChannel=all
 - It is a good idea to get a simultaneous MQ trace.

- Problem – There is a problem in a mediation handler, or the interaction between a mediation handler and a destination.
 - SIBMessageTrace=all:SIBMediations*=all:SIBMfp*=all
 - **Note:** There are also WebSphere Process Server mediations, check any messages in SystemOut for the package structure to see whether it is SIB or WebSphere Process Server.
 - You can tell easily by the package names.

Notes:

If the problem is not with the WebSphere Default Messaging mediation handlers, such as a WebSphere Enterprise Service Bus mediation, the package names in the stack will not include SIB.

Instructor notes:

Purpose — This section is to act as a guide for students to know what groups to use in what situation.

Details —

Additional information —

Transition statement — More what groups to use when.

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (5 of 6)

- Problem – You have applications connecting directly to a Q_Manager without going through the bus and something is not working properly.
 - SIBMessageTrace=all:SIBJms*=all
 - Get a simultaneous MQ Q_Manager trace.

- Problem – You are using a third party JMS provider and something is not right.
 - Message=all:SIBJms*=all
 - Message=all is a WebSphere Application Server trace group not owned by the messaging team.
 - It would also be useful to get the third party JMS provider support team involved to advise on any trace they may need to investigate the issue.

Notes:

Instructor notes:

Purpose — This section is to act as a guide for students to know what groups to use in what situation.

Details —

Additional information —

Transition statement — More what groups to use when.

Tracing for WebSphere Default Messaging – What to trace for specific types of problems (6 of 6)

- Problem – There is a problem involving an ME in a cluster not running on the preferred server, or not failing over properly.
 - SIBTrm=all:WLM*=all:Cluster*=all
- Problem - There is a problem with a MDB not putting or getting messages properly from the destination.
 - SIBMessage*=all:SIBProcessor=all:SIBJmsRa*=all:SIBRa*=all
 - If the destination is remote add SIBCommunications=all
- Problem – There is a problem configuring a default messaging object. The following trace setting cover both the administration console and wsadmin.
 - com.ibm.ws.management.*=all:com.ibm.websphere.management.*=all:com.ibm.ws.management.commands.sib.*=all:SIBAdmin=all:Webui=all
 - You can also investigate administration problem with the \$AdminConfig validate command.

Figure 17-33. Tracing for WebSphere Default Messaging – What to trace for specific types of problems (6 of 6)

WA5711.0

Notes:

There is a problem involving an ME in a cluster not running on the preferred server, or not failing over properly – These problems are best looked from a configuration perspective first using the information in the SystemOut.log and the XML configuration files.

There is a problem with a MDB not putting or getting messages properly from the destination. Be sure you are at or near the latest fix pack for these issues before tracing.

There is a problem configuring a WebSphere Default Messaging object: The XML configuration files are the first “port of call” for these types of problem as well. If problems like these get to trace, both the WebSphere Application Server Administration support team and the WebSphere Default Messaging support team will be involved.

Instructor notes:

Purpose — This section is to act as a guide for students to know what groups to use in what situation.

Details —

Additional information —

Transition statement — More what groups to use when.

Unit summary

Having completed this unit, you should be able to:

- Describe the components involved in default messaging
- Identify symptoms of default messaging related problems
- Collect diagnostic data
- Analyze diagnostic data and determine root causes
- Validate solutions

Figure 17-34. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 12: Troubleshooting the default messaging provider

Figure 17-35. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 18. WebSphere installation, update, and migration problems

Estimated time

01:00

What this unit is about

This module describes some common installation problems and how to troubleshoot them.

What you should be able to do

After completing this unit, you should be able to:

- Follow installation checklist to detect installation problems
- Locate and examine relevant installation logs
- Recover from a failed installation
- Use the installation verification utility
- Search for relevant version information and prerequisite levels

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

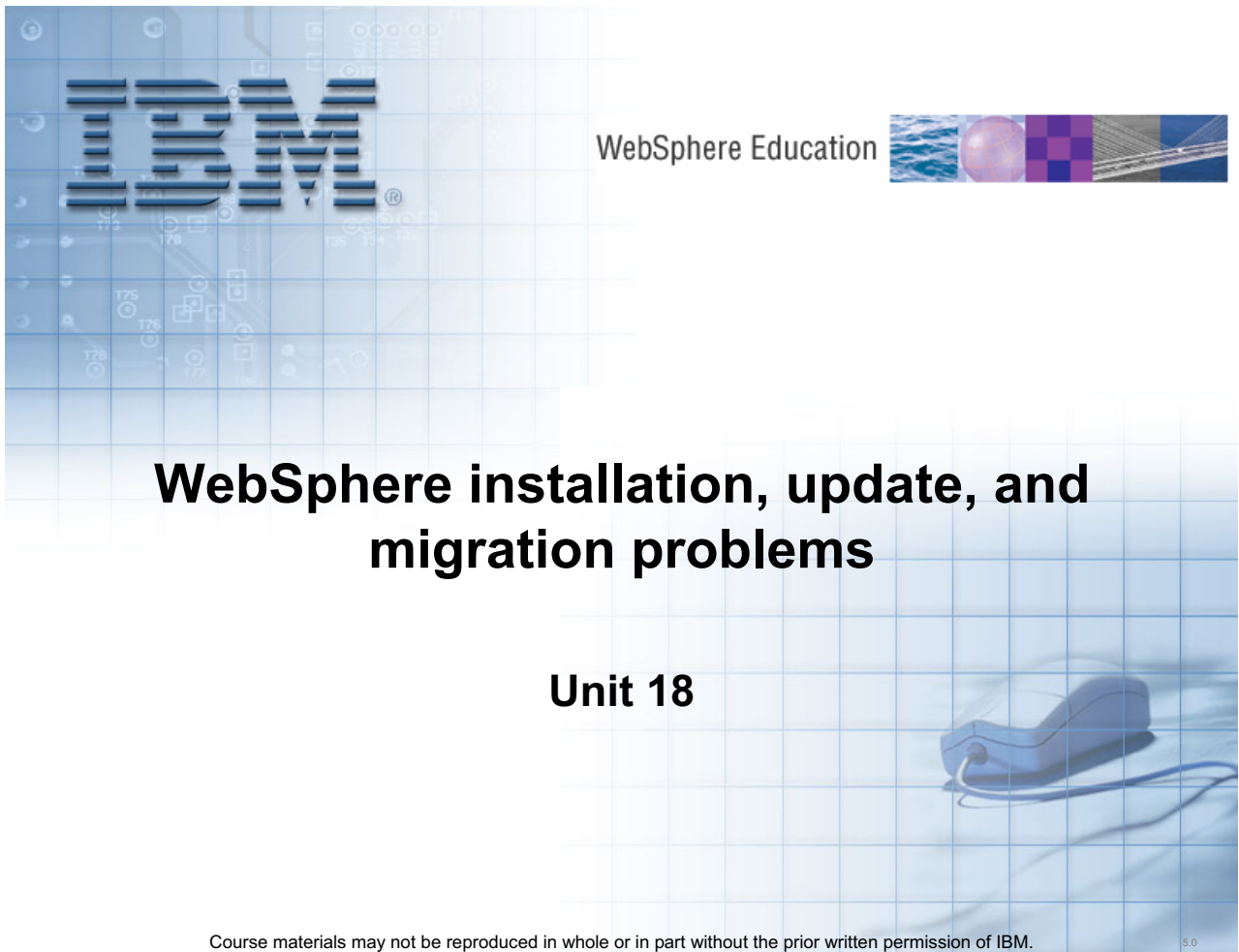
References

Red Paper: *WebSphere Application Server installation problem determination*

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4305.pdf>

Web page name: *WebSphere Application Server V6.1 information center*

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>



Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

5.0

Figure 18-1. WebSphere installation, update, and migration problems

WA5711.0

Notes:

Instructor notes:**Purpose —**

Details — This presentation provides a lot of detail on four major sources of installation problems: Launchpad failures, installation wizard failures, profile creation failures, and failure of the IVT. A common approach is used to present each case.

1. Identify the symptom
2. Recommend what log files to examine and data to collect
3. What to look for in the data

This is basically a problem determination approach rather than an attempt to solve all possible things that can go wrong during an installation.

Additional information —**Transition statement —**

Unit objectives

After completing this unit, you should be able to:

- Follow installation checklist to detect installation problems
- Locate and examine relevant installation logs
- Use the Installation Verification Tool (IVT)
- Use the Installation Verification Utility (IVU)
- Recover from failed installation—uninstall and clean machine
- Run install command with different options for logging, tracing and silent install
- Troubleshoot maintenance update installation
- Troubleshoot migration problems

Figure 18-2. Unit objectives

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Installing WebSphere Application Server

- A successful installation of WebSphere Application Server
 - Installs the core product files
 - Creates the server1 application server

- A successful installation of WebSphere Application Server ND is a two-part process:
 - First, use the installation wizard to install a shared set of core product files
 - Second, use the Profile Management Tool or *manageprofiles* command to create at least one profile
 - Cell (deployment manager plus federated application server profile)
 - Deployment manager profile
 - Application Server profile
 - Custom profile

Figure 18-3. Installing WebSphere Application Server

WA5711.0

Notes:

Federating nodes

Deploying an ND cell will also require federating nodes to the deployment manager. The process of federating nodes or adding nodes to a cell can be done by using either the `addNode` command or the Administrative Console. Some common problems encountered when federating remote nodes include:

1. Time synchronization issues between machines
2. Network connectivity

Other issues encountered are:

1. Out-of-memory error requiring Java heap size to be increased
2. Soap client timeout error requiring soap client timeout property to be increased

If the node federation process fails, the first place to look for detailed information is the `addNode.log` file which can be found in the logs directory for the node being federated.

Instructor notes:

Purpose — To begin with, establish what should happen when the installation process is successful.

Details —

Additional information —

Transition statement —

The launchpad for network deployment

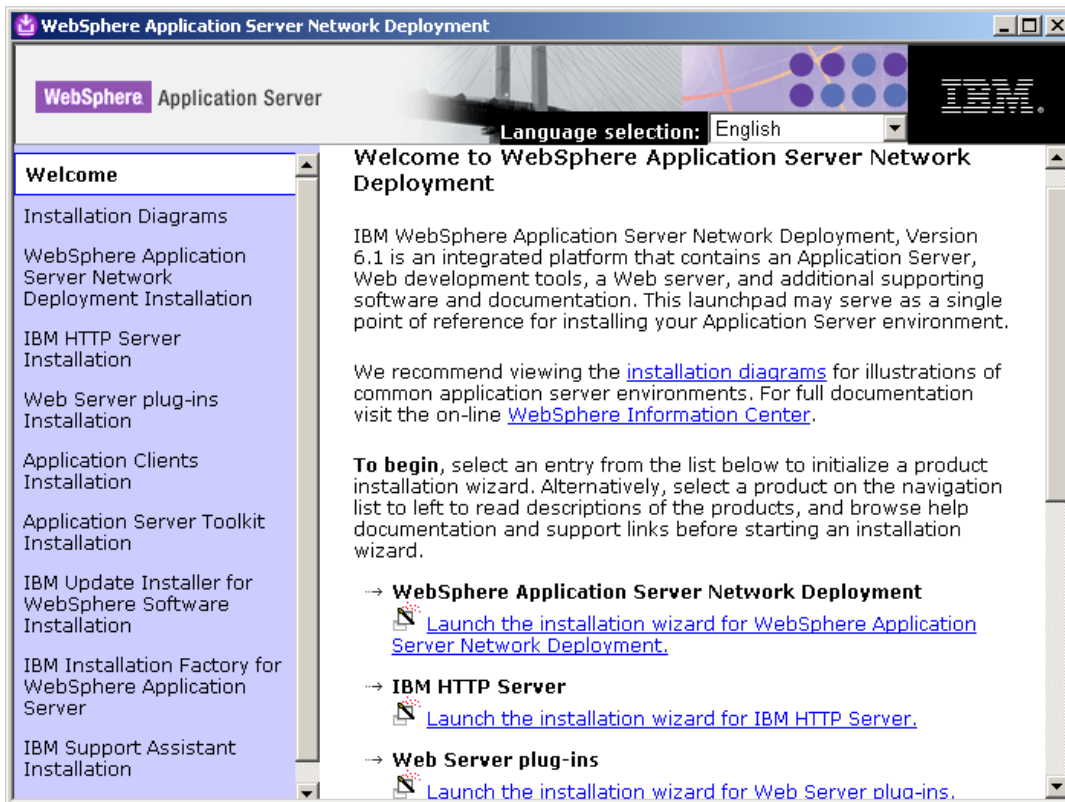


Figure 18-4. The launchpad for network deployment

WA5711.0

Notes:

Commands for silent installs

UNIX example:

```
install -options "/opt/IBM/WebSphere/silentFiles/myresponsefile.txt"
-silent
-log # !/opt/IBM/WebSphere/myOptionFiles/log.txt @ALL
```

Windows example:

```
install.exe -options "C:\IBM\WebSphere\silentFiles\myresponsefile.txt"
-silent
-log # !C:\IBM\WebSphere\silentFiles\log.txt @ALL
```

Instructor notes:

Purpose — Do not spend too much time on this slide. Just show the GUI for those students who may not have seen it.

Details —

Additional information —

Transition statement —

The Profile Management Tool

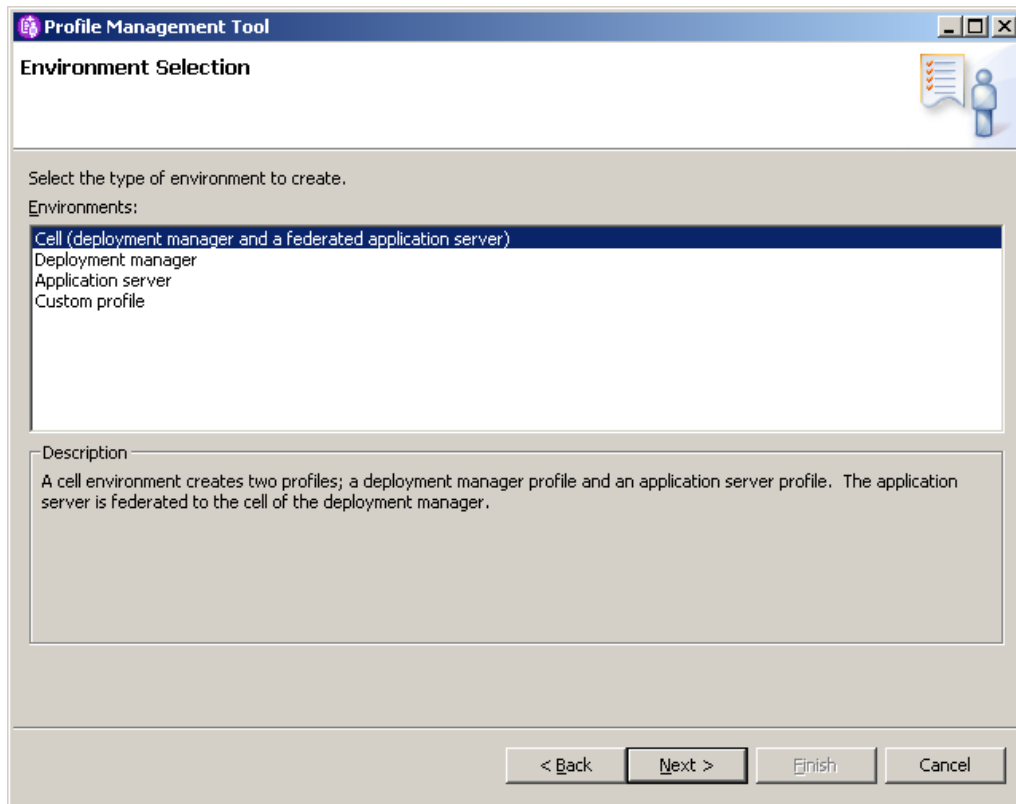


Figure 18-5. The Profile Management Tool

WA5711.0

Notes:

Example of creating a profile the command line in UNIX:

```
$ ./wasprofile.bat -create -profileName Node1 \
-templatePath "C:\was6\profileTemplates\default" \
-profilePath "c:\was6\profiles\Node1" \
-cellName Node1Cell \
-nodeName Node1 \
-hostName pvs4.pittsburgh.ibm.com \
-portsFile "C:\was6\temp\portdef.props"
```

Instructor notes:

Purpose — Do not spend too much time on this slide. Just show the GUI for those students who may not have seen it.

Details —

Additional information —

Transition statement —

What can go wrong

- Launchpad or installation wizard fails to start
- The installation process fails
- Profile creation fails
- The Installation Verification Test (IVT)
 - Fails to start the server
 - Fails to validate one or more components
- The Installation Verification Utility
 - Reports missing files
 - Reports changed files

Notes:

Instructor notes:

Purpose —

Details — Ask students to share their experiences with any installation failures they have encountered.

Additional information —

Transition statement —

Installation problem determination

- Evaluate the high-level symptoms to determine if one of them describes your problem
- If it does, collect the appropriate data that is required to diagnose the problem
- Next, research the documentation to determine where or what the problem is
- Take the next step for resolution
 - Access product support site
 - Contact IBM
 - Collect information about configuration
 - Some other recommended action
- **Note:** Make sure that you have read and met the hardware and software prerequisites

Figure 18-7. Installation problem determination

WA5711.0

Notes:

Before trying to determine the cause of an installation problem, make sure that you have read and met the hardware and software prerequisites. The latest information about these is available at <http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html>

Instructor notes:

Purpose — Just review a general approach to problem determination. The remainder of the presentation will address primarily the first two bullets. Recommend the use of the IBM Support Assistant for the last two bullets.

Details —

Additional information —

Transition statement —

Common installation problems

- Case 1: Launchpad or installation wizard will not start or fails
 - No supported browser, browser configuration
 - Insufficient disk-space
- Case 2: Installation wizard hangs
 - Low resources such as disk space, virtual memory
 - Problems with JVM
- Case 3: Profile creation failure
 - Long directory paths
 - File permissions
 - Host name
 - Conflicting ports
- Case 4: IVT fails
 - Server startup issues due to problems with port conflicts with other applications
 - Low resources such as application heap size

Figure 18-8. Common installation problems

WA5711.0

Notes:

Each of these cases will be described in detail on the following slides.

Instructor notes:

Purpose — Identify and briefly describe these four cases (high-level symptoms). The details of each case will be discussed in the slides that follow.

Details —

Additional information —

Transition statement —

Case 1: Launchpad or installation wizard start failure (1)

After completing this topic, you should be able to:

- Recognize symptoms of launchpad and install wizard failures
- Locate the relevant log files
- Collect problem determination data

Figure 18-9. Case 1: Launchpad or installation wizard start failure (1)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 1: Launchpad or installation wizard start failure (2)

- This case covers the following situations:
 - Launchpad failure
 - **launchpad.bat** command in Windows fails to start the launchpad
 - **launchpad.sh** script in UNIX fails start the launchpad
 - Installation wizard failures
 - Launchpad fails to start the installation wizard
 - The wizard exits with an error message
 - The wizard exits *without* an error message
- Possible causes for these problems include:
 - Web browser requirements not met
 - Insufficient disk space
 - Inadequate OS permissions

Notes:

Instructor notes:

Purpose — The next sequence of slides present a detailed discussion of Case 1. The general approach is:

1. Describe the symptoms
2. Discuss what data to collect and what logs to examine
3. Identify some issues that may be relevant to problem determination

Details —

Additional information — The students will work with the browser issues in the exercise that accompanies this unit.

Transition statement —

Case 1: Problems starting the launchpad

- Is a supported Web browser installed and configured properly?
 - Verify that the latest version of a supported Web browser is installed
 - Mozilla
 - Internet Explorer
 - On UNIX platforms, ensure that the location of the supported browser is exported
 - For example: export BROWSER=/usr/bin/mozilla
- Ensure that JavaScript is enabled in the browser options or preferences.
 - For example:
 - Mozilla for AIX: select **Edit -> Preferences -> Advanced -> Scripts & Plugins**
 - Enable JavaScript for: Navigator, Allow Scripts to: Select all
 - Internet Explorer: select **Tools -> Internet Options -> Security -> Custom Level for Internet Scripting -> Active scripting -> Enable**
 - Mozilla Firefox: select **Tools -> Options -> Content**
 - Select **Enable JavaScript** and click **Advanced**. Select all options

Figure 18-11. Case 1: Problems starting the launchpad

WA5711.0

Notes:

The launchpad is a Web application. Before using the launchpad, you must have a supported Web browser.

The launchpad supports the following browsers:

- Mozilla, Version 1.4 and 1.7.5 or later
- Internet Explorer, Version 5.5 with SP 2 or later

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 1: Problems with the installation wizard

- Installation events are logged in <WAS_install_root>/logs/install/log.txt

- If the installation wizard starts but fails during the installation process, look for messages that contain
 - INSTCONFSUCCESS
 - INSTCONFPARTIALSUCCESS
 - INSTCONFFAILED
 - These messages indicate the current status of the installation

- If you see the messages
 - INSTCONFPARTIALSUCCESS
 - INSTCONFFAILED
 - Look for error or warning messages preceding them that indicate problems with resources such as
 - Not enough disk space
 - Exceptions in the JVM
 - Segmentation faults
 - And so forth

Figure 18-12. Case 1: Problems with the installation wizard

WA5711.0

Notes:

If the log.txt file does not exist, then run the installer directly from a command window using **-log** option to create a log of all events.

Note that issuing the following commands allows you to start the installer directly, bypassing the launchpad. The install command is located in

<Product_CD_Dir>\C88SPML\WAS

- UNIX

```
./install -log !logfile @ALL
```

- Windows

```
install -log !logfile @ALL
```

Where *logfile* is a fully qualified file name for writing the log events.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Error: Suitable JVM could not be found

- An error that resembles the following indicates a problem with disk space:

```
A suitable JVM could not be found. Please run the program  
again using the option -is:javahome < JAVA HOME DIR> No space  
left on device
```

- This error indicates that there is not enough free space for the installer to run
 - This error is reported even if enough space exists where you plan to install WebSphere Application Server (for example, drive C: or /usr) .
- Verify that the location of %TEMP% in Windows or /tmp directory in UNIX has enough free space for the installer to run
- Or use -is:tempdir with the install command
 - Where tempdir is the location of a temporary directory on a partition with enough free space.

Figure 18-13. Error: Suitable JVM could not be found

WA5711.0

Notes:

Check the installation documentation to determine the exact amount of temporary disk space required; typically it is a minimum of 100 MB.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 2: Installation wizard hangs (1)

After completing this topic, you should be able to:

- Recognize symptoms of installation wizard hangs
- Locate the relevant log files
- Collect problem determination data

Figure 18-14. Case 2: Installation wizard hangs (1)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 2: Installation wizard hangs (2)

- During installation, a progress indicator is displayed to show how far the install has progressed
- If there is no change in the progress indicator for a very long time (>10 minutes), the installation process could be hung
- Reasons that the install process might hang include the following:
 - The system is very low on resources such as virtual memory or swap space
 - There might be heavy network traffic or network breakdown if installing from a remote location
 - A task or thread has gone into an infinite wait or loop

Figure 18-15. Case 2: Installation wizard hangs (2)

WA5711.0

Notes:

When running an install wizard it is helpful to use Task Manager on Windows to monitor CPU utilization. This way, even if the progress indicator seems hung, you can still determine if any processing is going on.

On UNIX machines, it is helpful to have a window open where you are running the command: `tail -f log.txt`

Instructor notes:

Purpose — The next sequence of slides present a detailed discussion of Case 2. The general approach is:

1. Describe the symptoms
2. Discuss what data to collect and what logs to examine
3. Identify some issues that may be relevant to problem determination

Details —

Additional information —

Transition statement —

Case 2: Data to collect for installation hangs

- If the installation hangs, check
 - <WAS_install_root>/logs/install/log.txt
 - Logs all installation events
 - Return codes: 0=success, 1=failure, 2=partial success

- If the installer fails at a very early stage, then this log file might not be created or it might exist in the system temporary area, which is:
 - Windows: %TEMP%\log.txt
 - UNIX: /tmp/log.txt

- Other log in the <WAS_install_root>/logs/install directory include:
 - installconfig.log
 - Logs the activities of ANT configuration scripts that run at the end of the installation procedure
 - trace.txt

Figure 18-16. Case 2: Data to collect for installation hangs

WA5711.0

Notes:

Instructor notes:

Purpose — Just identify the logs and their location.

Details —

Additional information —

Transition statement —

Case 2: What to look for if installation hangs

- If you think the installation process is hung, check the *log.txt* and *installconfig.log* files periodically to see if progress is being made
- Check for status messages such as:
 - INSTCONFSUCCESS
 - INSTCONFPARTIALSUCCESS
 - INSTCONFFAILED
- If you see the INSTCONFPARTIALSUCCESS or INSTCONFFAILED messages, then look for *error* or *warning* messages preceding them
 - check other system activities such as:
 - CPU utilization
 - Hard disk usage
 - Or any network activity (if installing remotely)
- If the installation *does* appear to be hung, look for the last recorded message in the log file
 - This message gives you an idea of what the installer was doing before it hung

Notes:

Instructor notes:

Purpose —

Details —

Additional information — In the exercise, students will examine several logs and search for the status messages discussed in this slide.

Transition statement —

Case 2: Identify which process failed (1 of 2)

- The installation processes are as follows:
 - Prerequisites check
 - Check system for a supported operating system
 - Check operating system for appropriate service packs and patches
 - After prerequisites check, check for existing WebSphere Application Server products
 - Install core product files
 - Create installation directory
 - Extract contents of install media
 - Create uninstaller
 - Copy files from installation media to installation directory (file copy process)
 - Configure components

Figure 18-18. Case 2: Identify which process failed (1 of 2)

WA5711.0

Notes:

The prerequisites checker process will write information to the log file `%TEMP%\IBM_WebSphere_launchpad\waspc_check<xxxxxx>out`. If your system fails the prerequisite check, examine this log file for more details.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 2: Identify which process failed (2 of 2)

- Determine if errors occurred during the *file copy and configuration process*
 - Look in *log.txt* for an entry such as the following:

```
(<Date and time stamp>), Install,  
com.ibm.ws.install.ni.ismp.actions.  
ISMPConfigManagerLaunchAction, msg1, INSTCONFSUCCESS:  
Post-installation configuration is successful
```

- If you see this message, the file copy and configuration process has completed successfully
- If not, inspect the messages in the log for an indication of the error

Notes:

Instructor notes:

Purpose —

Details — Point out that the **ISMPConfigManagerLaunchAction** value in the log entry refers to the file copy and configuration action.

Additional information —

Transition statement —

Case 3: Profile creation failure

After completing this topic, you should be able to:

- Recognize symptoms of profile creation failures
- Locate the relevant log files
- Collect problem determination data

Figure 18-20. Case 3: Profile creation failure

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 3: Profile creation failure

- Problems with profile creation might be due to:
 - Long directory paths
 - File permissions
 - Problems with the host name (using “localhost”)
- A default server profile is created as part of the installation process in a WebSphere Application Server or WebSphere Application Server – Express installation
- The Network Deployment installation wizard gives you the option of creating a profile
 - Profiles can also be created at any time after installation.

Notes:

Profiles allow you to define multiple runtime environments, each with its own administrative interface while sharing the same code base.

Instructor notes:

Purpose — The next sequence of slides present a detailed discussion of Case 3. The general approach is:

1. Describe the symptoms
2. Discuss what data to collect and what logs to examine
3. Identify some issues that may be relevant to problem determination.

Details —

Additional information —

Transition statement —

Case 3: Verify that profile creation succeeded

- Determine if the errors occurred in the *profile creation process*
 - If the *file copy and configuration process* succeeded, any error messages after this indicate problems in profile creation or later steps, including
 - Sample application deployment
 - Administrative console application deployment
- Look for the following entry in the log

```
(<Date and time stamp>),Install,  
com.ibm.ws.install.ni.ismp.actions.  
ISMPWSProfileLaunchAction, msg1,INSTCONFSUCCESS:  
Post-installation configuration is successful
```

- If you do *not* see this message, then there are problems with profile creation

Figure 18-22. Case 3: Verify that profile creation succeeded

WA5711.0

Notes:

The log file referred to here is:

- `<Install_Root>log.txt` for WebSphere Application Server and WebSphere Application Server - Express
- `<WAS_install_root>/profiles/<profile>/logs/pctLog.txt` for WebSphere Application Server Network Deployment

Instructor notes:

Purpose —

Details — Point out that the **ISMPWSProfileLaunchAction** value in the log entry refers to the profile creating and configuration action.

Additional information —

Transition statement —

Case 3: Profile creation steps

- Profile creation steps are as follows:
 - Create directory structure
`<WAS_install_root>/profiles/<profile>/`
 - Configure profile using one of the templates from `<WAS_install_root>/profileTemplates`
 - cell
 - default
 - dmgr
 - managed

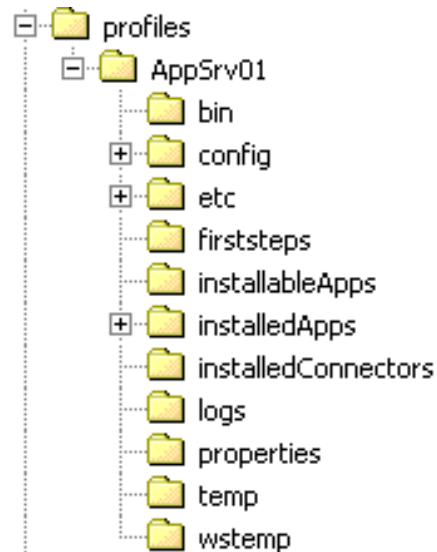


Figure 18-23. Case 3: Profile creation steps

WA5711.0

Notes:

Profile templates consists of XML configuration files, properties files, ant tasks, and so on.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 3: Data to collect if profile creation fails (1 of 3)

- If the profile creation fails, check
`<WAS_install_root>/logs/manageprofiles/<profile>_create.log`
- This log file is created when the file copy process has completed and the creation of a default profile starts
 - also created when **manageprofiles** command is executed
 - traces all the events that occur during profile creation
- It is an XML log file and can be viewed with
 - Web browser
 - WordPad in Windows
 - Log Analyzer (ISA and AST)
- The entries in this log consist of `<record>... </record >` stanzas

Figure 18-24. Case 3: Data to collect if profile creation fails (1 of 3)

WA5711.0

Notes:

A sample log entry that indicates an error would look something similar to the following:

```
<record>
<date>2005-07-19T14:18:53</date>
<millis>1121762933500</millis>
<sequence>2984</sequence>
<logger>com.ibm.ws.install.configmanager.ConfigManager</logger>
<level>WARNING</level>
<class>com.ibm.ws.install.configmanager.ConfigManager</class>
<method>executeAllActionsFound</method>
<thread>11</thread>
<message>Fatal configuration action failed:
com.ibm.ws.install.configmanager.actionengine.ANTAction
-C:\IBM\WAS\profileTemplates\managed\actions\
executeManagedProfileSetup.ant</message>
</record>
```

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 3: Data to collect if profile creation fails (2 of 3)

- Numerous log files are created when a profile is created in the directory `<WAS_install_root>/logs/manageprofiles/<profile>`
- Logs created depend on the type of profile (deployment manager, application server, custom)
- Some of these logs for the deployment manager include
 - collect_metadata.log
 - createDefaultServer.log
 - defaultapp_config.log
 - Service.log
 - filetransfer_config.log (for the file transfer application)
 - ivt_config.log for the (install verification application)
 - SIBDefineChains.log
 - SIBDeployRA.log
- **Note:** the `portdef.props` file maybe useful when resolving port conflicts

Figure 18-25. Case 3: Data to collect if profile creation fails (2 of 3)

WA5711.0

Notes:

Each log might or might not exist depending on the type of profile that is created.

- pctLog.txt: Created only when the profile creation wizard is executed. This log is not created when using the **wasprofile** command directly or during installation of the product.
- amjrte_config.log: Tivoli Access Manager configuration log for its Java Runtime Environment.
- collect_metadata.log: Collects metadata information about managed objects in the system to evaluate and prevent potential installation conflicts.
- createDefaultServer.log: A log from wsadmin recording the creation of the server1 process in the default profile.
- createshortcutforprofile.log: Windows tool log for creating menu entries and shortcuts.
- defaultapp_config.log: JACL script log from configuring default application resources.
- Service.log: Start and stop events for server1.

Application installation and configuration logs for system and sample applications:

- filetransfer_config.log for the file transfer application
- ivt_config.log for the ivtAPP application
- mejb_config.log for the ManagementEJB application
- hamanager_config.log for the high availability application
- query_config.log for the Query application
- samples_config.log for the PlantsByWebSphere sample application
- samples_install.log for the SamplesGallery and PlantsByWebSphere sample applications
- scheduler.cal_config.log for the SchedulerCalendars application
- webui_config.log for the administrative console application
- defaultapp_deploy.log for the DefaultApplication application

SIBDefineChains.log: Creation log for Service Integration Bus endpoints, inbound channels and channel chains, outbound thread pool, and outbound channel and channel chains.

- SIBDeployRA.log: Deployment log for the Service Integration Bus function.
- winservice_config.log: Service log for the Windows service created for server1.
- addNode.log: Log for federating a node to a cell (custom profile).

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 3: Data to collect if profile creation fails (3 of 3)

- Each profile has a portdef.props file located at
 - <WAS_install_root>\logs\manageprofiles\
- Examine this file if you have a port conflict

```
#Generated by PMT GUI
#Tue Apr 03 15:48:22 EDT 2007
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=9403
WC_adminhost=9060
DCS_UNICAST_ADDRESS=9352
BOOTSTRAP_ADDRESS=9809
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=9401
CELL_DISCOVERY_ADDRESS=7277
SOAP_CONNECTOR_ADDRESS=8879
ORB_LISTENER_ADDRESS=9100
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=9402
WC_adminhost_secure=9043
```

Figure 18-26. Case 3: Data to collect if profile creation fails (3 of 3)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 3: What to look for if profile creation fails (1 of 2)

- Check for status messages in
 - `<WAS_install_root>/logs/manageprofiles/<profile>_create.log`
- Look for the following entries:
 - `<level>WARNING</level>`
 - `<level>SEVERE</level>`
 - `<message> INSTCONFPARTIALSUCCESS`
 - `<message> INSTCONFFAILED`
 - `<message>text - FAILURE </message>`
- If you see the `INSTCONFPARTIALSUCCESS` or `INSTCONFFAILED` messages, then look for *error* or *warning* messages preceding them
- If you do *not* see an entry similar to the following, then there are problems with profile creation:

```
(<Date and time stamp>),  
Install,com.ibm.ws.install.ni.ismp.actions.  
ISMPWSPProfileLaunchAction, msg1,INSTCONFSUCCESS:  
Post-installation configuration is successful
```

Figure 18-27. Case 3: What to look for if profile creation fails (1 of 2)

WA5711.0

Notes:

Most tasks, such as system or sample application installation, are logged to individual log files in the `<WAS_install_root>/profiles/<profile>/logs` directory. If you can determine which task the profile creation was doing, collect the file for that task.

For example, if you had problems creating a custom node then look at the log file `<WAS_install_root>/profiles/<profile>/logs/addNode.log` for any errors.

Instructor notes:

Purpose —

Details —

Additional information — In the exercise, students will examine several logs and search for the status messages discussed in this slide.

Transition statement —

Case 3: What to look for if profile creation fails (2 of 2)

- Look for an entry such as the following:

```
(<Date and time stamp>), Install,
com.ibm.ws.install.ni.ismp.actions.
ISMPWSProfileLaunchAction, err,INSTCONFFAILED: Cannot
complete required configuration actions after
installation. The configuration failed. The
installation is not successful.Refer
to\install_root\logs\wasprofile\wasprofile_create_pro
filename.log for more details
```

- If you see this entry, examine the log and try to determine what task was being performed when the profile creation failed

Figure 18-28. Case 3: What to look for if profile creation fails (2 of 2)

WA5711.0

Notes:

The following are common problems with profile creation.

File path length errors

Windows 2000 has a length restriction of 258 characters for a command. A problem can occur that prevents the successful creation of a profile when a path is too long. The maximum length for the `<WAS_install_root>` directory is 60 characters. The maximum length for the profiles installation root directory is 80 characters.

If your log files have errors similar to Input line is too long then the length of the file path and node name in the command string has caused the entire command to exceed the operating system limit for command length. This error can show up in a message box during the wizard or if using wasprofile, in

`<WAS_install_root>/profiles/<profile>/logs/collect_metadata.log`. Create the profile again using shorter directory paths and node names. If you are still in the installation process, consider reinstalling using a shorter path for the installation root.

Host name error

If you see errors similar to localhost is not a valid host name for remote access, then you have entered localhost as the value in the host name field of the Profile creation wizard. Other machines in the network cannot reach your node using localhost, so you must provide a host name that can be resolved by other systems to the IP address of your system.

Template path error

If in a UNIX environment you get an error similar to the following then verify the profile templates location and permissions are correct:

```
<message>Incoming command line is: { "-create" , "-help"
, "-templatesPath", "/opt/WebSphere/AppServer/crso/profileTemplates/managed"}
</message>
<message>Could not resolve templatePath from command line</message>
```

Custom profile error

If an error occurred creating a custom profile, look at the addNode.log file for any additional errors. Look for an error message similar to the following:

```
[Date and Time stamp] 0000000a AbstractNodeC E ADMU0006E: Exception creating
Deployment Manager connection:
com.ibm.websphere.management.exception.ConnectorException: ADMC0016E: The
system cannot create a SOAP connector to connect to host hostname at port
8879
```

If you see this message, it is likely that the deployment manager profile is not running. Ensure that the deployment manager profile is created and running on the specified port and host before creation of custom profile.

Instructor notes:

Purpose —

Details —

Additional information — In the exercise, students will troubleshoot a profile creation problem.

Transition statement —

Use ISA to search for profile creation problems

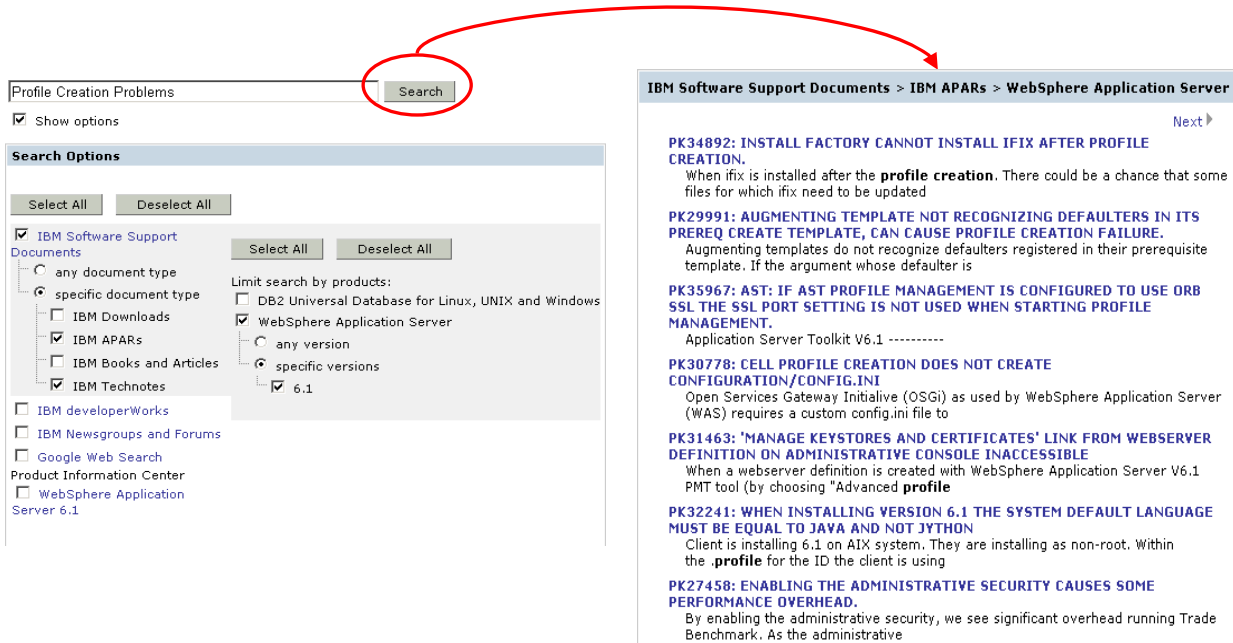


Figure 18-29. Use ISA to search for profile creation problems

WA5711.0

Notes:

Instructor notes:

Purpose — Use as an example of searching with ISA. Point out how narrowing the search (on the left, selecting APARs and choosing WebSphere Application Server V6.x) produces the results returned (on the right)

Details — Note that on the slide there is info about PK17944. This scenario is used in the exercise to create a profile creation failure.

Additional information —

Transition statement —

Case 4: Installation Verification Tool

After completing this topic, you should be able to:

- Recognize symptoms of IVT failures
- Locate the relevant log files
- Collect problem determination data

Figure 18-30. Case 4: Installation Verification Tool

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 4: Installation Verification Tool (1 of 2)

- The IVT looks at the profile configuration for the server and looks for a server running on the server port number
 - Run from **First steps** wizard
 - <WAS_ROOT>\profiles\<profile_name>\firststeps\firststeps.bat/sh
 - From the command line (ivt.bat/sh)
 - Example: `ivt.bat server1 profile01 -p 9080 -host was61host01`
- Note that if a server is up and running on the port, the IVT runs against that server
- If no server is running on that port, the IVT attempts to start the server
- When the server has started successfully, the IVT accesses the server and runs various tests, including:
 - Servlet engine verification
 - JSP verification
 - EJB verification
 - And others

Figure 18-31. Case 4: Installation Verification Tool (1 of 2)

WA5711.0

Notes:

Format for running IVT from the command line:

```
Ivt.bat/sh <SERVER_NAME> <PROFILE_NAME> [-p port] [-h hostname]
```

Instructor notes:

Purpose — The next sequence of slides present a detailed discussion of Case 4. The general approach is:

1. Describe the symptoms
2. Discuss what data to collect and what logs to examine
3. Identify some issues that may be relevant to problem determination

Details —

Additional information —

Transition statement —

Case 4: Installation Verification Tool (2 of 2)

- You should run the IVT before making any configuration changes to WebSphere Application Server
- This acts as a checkpoint to see if any problems exist as a result of the installation
- If the IVT runs clean and problems show up later, they are most likely due to configuration changes done *after* the installation
- Stop all instances of WebSphere Application Server before running the IVT
- IVT fails usually because the application server fails to start
 - Common reasons for this include:
 - Port conflicts
 - Not enough memory

Notes:

Instructor notes:

Purpose —

Details —

Additional information — In the exercise, students use the IVT from the First Step console and also from the command line. A failure scenario from the IVT has not been included in the exercise.

Transition statement —

Case 4: Data to collect if IVT fails

- The logs most likely to be of interest are:
 - `<WAS_install_root>/profiles/<profile>/logs/ivtClient.log`
 - This log contains messages from the IVT execution

 - The following log files are created in the directory:
 - `<WAS_install_root>/profiles/<profile>/logs/server1`
 - startServer.log - Log of start server events
 - SystemOut.log - Log of all activity within the WebSphere environment
 - SystemErr.log - Record of system errors

Notes:

The JVM files for server1 (or any server you run IVT against) should be examined if the server fails to start.

Instructor notes:

Purpose —

Details — Point out that the SystemOut.log is most likely to contain specific information about why a server failed to start.

Additional information —

Transition statement —

Case 4: What to look for if IVT fails (1 of 3)

- The first file to look at in case IVT has failed is *ivtClient.log*
 - Messages with IVTL and ADMU prefixes provide information about what the IVT application is doing and the status of each action
- Look for the following message in *ivtClient.log*:

```
ADMUXXXXX: Server servername open for e-business;  
process id is xxxx
```
- If you find this message, then:
 - The server has started successfully
 - But the IVT failed while executing one of the tests
- Examine the error messages in *ivtClient.log* to locate the failing process

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 4: What to look for if IVT fails (2 of 3)

- If another deployment manager is running with the same port, you see the following messages:

```
IVTL0110E: Log file error with  
C:\WebSphere\AppServer\profiles\Dmgr02\logs\dmgr\  
SystemOut.log, java.io.FileNotFoundException:  
C:\WebSphere\AppServer
```

- If the server does not start, look at the `Server Port number is:` entry.
 - This entry contains the server port number of the profile instance
 - Make sure that this port is not in use

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Case 4: What to look for if IVT fails (3 of 3)

- If the port does not seem to be the problem, look at
 - startServer.log
 - SystemErr.log
 - SystemOut.log
- Look for information related to server startup in the startServer.log and SystemOut.log, such as the following:
 - Any error messages starting with WSVR (server runtime)
 - ADMU (management utility).
- Look for error messages in SystemErr.log These messages will start with:
 - [Date and time stamp] 0000000a SystemErr

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Installation Verification Utility

After completing this topic, you should be able to:

- Explain how the IVU is used to verify the integrity of installed files
- Use the IVT to verify installations of the Application Server, deployment manager, Application Client, IBM HTTP Server, Web server plug-ins, Update Installer, and Fix Packs

Figure 18-37. Installation Verification Utility

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information — Make sure the students understand that the IVT and IVU are totally different verification tools.

Transition statement —

Installation Verification Utility: installver

- Installation Verification Utility (IVU) can be used to detect and diagnose initial installation issues
 - Computes the actual checksum value for the installed files and compares them to the shipped bill-of-materials list (several `files.list` files)
 - Reports missing or changed files
- After installing a refresh pack/fix pack, the bill-of-materials is automatically updated
- IVU can be used for verifying
 - Application server/deployment manager
 - Application client
 - IBM HTTP Server
 - Web server plug-ins
 - Update installer
 - Refresh pack/fix pack
- Use IVU to create a new checksum after configuring or reconfiguring the system
 - Stores the new checksum in the `sys.inv` file within the current working directory

Figure 18-38. Installation Verification Utility: installver

WA5711.0

Notes:

See the WebSphere Application Server V6.1 Information Center article at http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/tins_installver.html

A new install verification tool was introduced in V6.0.2, called the Install Verification Utility or “IVU” (the actual command name is “installver”). This tool uses checksums to verify that the product files have not been altered since they were installed.



Note

Do not confuse with the Installation Verification Tool (IVT) application which is available from the First Steps console to validate that the server is functioning properly. This is very different from the file checksum verification provided by IVU.

Use the `installver -help` for detailed usage information such as:

- `-help`
Help. Use this option to display this help information.
- `-trace`
Trace. Use this option to enable tracing. Trace output can be verbose. Combine the trace option with the log file option to direct verbose output to a file.
- `-log [someDir\logFile]`
Log. Use this option to enable the log file and redirect console output to the file. The default log file is `C:\Program Files\IBM\WebSphere\AppServer\logs\installver.log`. If a log file exists, output is appended.
- `-createtemplate`
Create the template file. Use this option to create a template file in your profile. A template provides the utility with specific actions against specific files. Currently, the template supports only file exclusions.
- `-createinventory someDir\sys.inv`
Create inventory. Use this option to create an inventory file. The default file is the `sys.inv` file in the current working directory, which is usually the bin directory of the profile. You can specify a different name and a different file path.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Run the verifyinstallver command

- `<WAS_install_root>/bin>verifyinstallver`

```
I CWNVU0160I: [ivu] Verifying.
I CWNVU0170I: [ivu] The installation root directory is C:\WebSphere\AppServer\
I CWNVU0300I: [ivu] The total number of user excluded files found is 0.
I CWNVU0300I: [ivu] The total number of IBM excluded files found is 65.
I CWNVU0185I: [ivu] Searching component directory for file listing: files.list
I CWNVU0460I: [ivu] The utility is running.
I CWNVU0260I: [ivu] The total number of components found is: 269
I CWNVU0270I: [ivu] Gathering installation root data.
I CWNVU0290I: [ivu] Starting the verification for 2 components.
I CWNVU0470I: [ivu] Starting to analyze: installver
I CWNVU0480I: [ivu] Done analyzing: installver
I CWNVU0470I: [ivu] Starting to analyze: installver.bin
I CWNVU0480I: [ivu] Done analyzing: installver.bin
I CWNVU0400I: [ivu] Total issues found : 0
I CWNVU0340I: [ivu] Done.
```

Figure 18-39. Run the verifyinstallver command

WA5711.0

Notes:

Use the **verifyinstallver** command to perform a checksum on the files that comprise the **installver** command.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Running the installver utility

- Run from `<WAS_install_root>/bin`
 - The default log file is the `<WAS_install_root>/logs/installver.log` file
 - Redirect the output using the `-log` parameter and an argument.
 - Use the `-log` parameter without the file argument to generate the default log file
 - Example: `installver -log`

```
I CWNVU0470I: [ivu] Starting to analyze: nif.componentmap.embed
I CWNVU0480I: [ivu] Done analyzing: nif.componentmap.embed

I CWNVU0470I: [ivu] Starting to analyze: nif.componentmap.embed.common
I CWNVU0440I: [ivu] The following file is different:
properties/version/nif/componentmaps/componentmap.coreruntime.embed.common.xml
I CWNVU0410I: [ivu] 74e081bcfb5db1d0e4a19682302aac84d34d24ce is the checksum
in the bill of materials.
I CWNVU0420I: [ivu] c10bf99e87829af2549eee72c902c4dea6cf34c3 is the checksum
on the file system.
I CWNVU0390I: [ivu] Component issues found : 1
I CWNVU0480I: [ivu] Done analyzing: nif.componentmap.embed.common
```

Figure 18-40. Running the installver utility

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Creating a new baseline checksum

- After configuring the product, create a new baseline checksum to establish a new checksum standard for your system
 - Example: `installver -createinventory`

```
C:\WASv61\bin>installver -createinventory
W CWNVU0320W: [ivu] The C:\WASv61\bin\sys.inv inventory file is within the product
installation root directory: C:\WASv61.
I CWNVU0300I: [ivu]The total number of user excluded files found is 0.
I CWNVU0300I: [ivu]The total number of IBM excluded files found is 82.
I CWNVU0310I: [ivu]Creating the following inventory file: C:\WASv61\bin\sys.inv
I CWNVU0460I: [ivu] The utility is running.
...
I CWNVU0340I: [ivu] Done.
```

- Later, compare the checksums in the `sys.inv` file to the checksums of the currently installed files to see what files have changed

Figure 18-41. Creating a new baseline checksum

WA5711.0

Notes:

Computing a new baseline checksum for an inventory of configured files

After installation, you can verify the actual checksums of installed files against a bill of materials that ships with the product. After configuring your system, create a new checksum so that you can compare the system periodically to the new checksum. Use the result to analyze changes to your configured system.

After configuring the product, save a new baseline checksum to establish a new checksum standard for your system.

Use the **installver** command to create and compare an inventory of configured files to the currently installed files.

The **installver** tool can compute a new baseline checksum for the inventory of all files in the installation root directory. The tool stores the new checksum by default in the `sys.inv` file within the current working directory. You can specify a different file path and file name. Create the file outside of the installation root directory or exclude the file from comparisons.

Later, compare the checksums in the sys.inv file (or the file that you specified when creating the inventory) to the checksums of the currently installed files to see what files have changed.

The baseline checksum report identifies missing files, additional files, and changed files.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Uninstalling after failed or hung install

After completing this topic, you should be able to:

- Recover from a failed installation

Figure 18-42. Uninstalling after failed or hung install

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Recovering from a failed or hung installation (1 of 2)

- Depending on the state of your system when an installation fails or hangs, you might need to uninstall WebSphere Application Server manually before you retry the process
- The uninstaller program leaves some files that can prevent you from reinstalling into the original directory
 - Delete the installation root directory and any registry entries to *clean* the machine so that you can reinstall into any directory
- Cleaning the machine means deleting everything from the previous installation,
 - Including *log files* that are left behind by the uninstall command
 - Before you start the procedure, back up log files, if necessary
- Manually uninstalling produces a clean system
 - A clean system has no evidence of a previously deleted installation

Notes:

Instructor notes:

Purpose — Provide general steps for uninstalling WebSphere Application Server.

Details —

Additional information — The students will do an uninstall as part of the exercise.

Transition statement —

Recovering from a failed or hung installation (2 of 2)

- If possible, issue the uninstall command:
`<app_server_root>_uninst\uninstall`
- If the install fails/hangs before the uninstaller is created, the process of uninstalling is more complex including the following steps:
 - Backing up and editing the registry
 - Deleting the installation root directory
 - Editing/deleting the *vpd.properties* file
- Refer to the Information Center for documented procedures
- Platform-specific instructions can be found at:
 - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_uninstall.html

Figure 18-44. Recovering from a failed or hung installation (2 of 2)

WA5711.0

Notes:

Refer to the following information center articles for manually uninstalling on different platforms.

Manually uninstalling on:

- AIX system
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_uninstallUnix.html
- HP-UX system
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_uninstallHpux.html
- Linux system
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_uninstallLinux.html

- Solaris system

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tins_uninstallSolaris.html

Manually uninstalling on Windows

Perform the following procedure to produce a clean system.

1. Log on with the same user ID that you used to install the product.
2. Run the uninstaller program for the Web server plug-ins for WebSphere Application Server. If a Web server is configured to run with the application server, uninstall the plug-ins to remove the configuration from the Web server.
3. Stop any browsers and any Java processes related to WebSphere Application Server products.
4. Issue the uninstall command. If you have already run the uninstaller program or if you cannot run the uninstaller program, skip this step.

```
app_server_root\uninstall\uninstall
```

The Uninstaller wizard begins and displays the Welcome panel.

5. Verify that you have an Emergency Recovery Disk. Instructions for creating this disk are in the Windows help documentation. This step is a safeguard. This procedure does not require the recovery disk.
6. Use the regback.exe program from the Windows Resource Kit to back up the registry. This step is a safeguard. This procedure does not require the backup copy of the registry.
7. Delete product registry entries for the WebSphere Application Server product that you are uninstalling. Edit the Windows system registry by invoking the regedit.exe command from a command prompt.



Important

Handle the registry with care. You can easily make a mistake while using the Registry Editor to view and edit registry contents. The editor does not warn you of editing errors, which can be extremely dangerous. A corrupt registry can disrupt your system to the point where your only option is to reinstall the Windows operating system.

8. Press Ctrl+F to search for all instances of WebSphere to determine whether you should delete each entry. You might not be able to remove all of the entries related to WebSphere Application Server, which is not a problem.

9. Expand and select keys related to WebSphere Application Server products. For example:
HKEY_CURRENT_USER\SOFTWARE\IBM\WebSphere Application Server Network Deployment\6.1.0.0
See the information center for a list of Windows registry keys to search for and delete.
10. Click **Edit -> Delete** from the menu bar for each related key.
11. Click **Yes** when asked to confirm deletion of the key.
12. Click **Registry -> Exit** from the menu bar when you are finished.
13. Delete the installation root directory for the product that you are uninstalling.
14. Determine all of the profile directories and remove the directories.
15. Open a Windows Explorer window and browse to the C:\Documents and Settings\All Users\Start Menu\Programs\IBM WebSphere directory. If you have one installation of a WebSphere Application Server product, delete the Application Server Network Deployment v6.x folder.
16. Delete the %WINDIR%\IsUninst.exe file. %WINDIR% is a variable. The Windows installation folder might be, for example, C:\Windows or C:\Winnt.
17. Edit the vpd.properties file. The file is located in the installation directory of the operating system, such as the C:\WINNT directory or the C:\Windows directory.
Do not delete or rename the vpd.properties file because the InstallShield MultiPlatform (ISMP) program uses it for other products that it installs. If the WebSphere Application Server product that you are uninstalling is the only product with entries in the vpd.properties file, you can delete this file.
18. Use the installRegistryUtils command to examine the installation locations for all installed WebSphere Application Server products and remove the desired products from the install registry.
19. Restart your machine if a prompt displays that directs you to restart.

Instructor notes:

Purpose —

Details —

Additional information — Recommend that students read all relevant documentation before attempting to uninstall.

Transition statement —

Reinstall with tracing turned on

- If you can successfully perform an uninstall, try reinstalling from the command line
 - install –is:javaconsole
 - Reports the stdout and stderr logs to the console window
 - Install –is:javaconsole > C:\capturefilename.txt
 - Captures the stream to a file
 - install –W Setup.product.install.logAllEvents="true"
 - Turns on additional installation logging

- Perform a silent install
 - install -options "C:\temp\responsefile.txt" -silent -log # !C:\temp\log.txt @ALL

Figure 18-45. Reinstall with tracing turned on

WA5711.0

Notes:

A sample response file called *responsefile.nd.txt* can be found in the directory `<Product_CD_Dir>\C88SPML\WAS`

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Applying maintenance updates

After completing this topic, you should be able to:

- Verify product version level
- Install Update Installer
- Install update: interim fix, fix pack, refresh pack
- Check update installation logs
- Use backupConfig and restoreConfig commands

Figure 18-46. Applying maintenance updates

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Verify current version levels and applied updates

- Use the following tools from `<WAS_ROOT>bin`
 - `versionInfo.bat/sh` [`-long`, `-maintenancePackages`, `-maintenancePackageDetail`, ...]
 - Shows build version, maintenance packages installed/uninstalled
 - `historyInfo.bat/sh` [`-maintenancePackageID ID_of_maintenance_package`]
 - Pinpoint specific interim fix, fix pack, or refresh pack information
- From the administrative console, select **Application servers -> server_name -> Runtime tab -> Product Information**

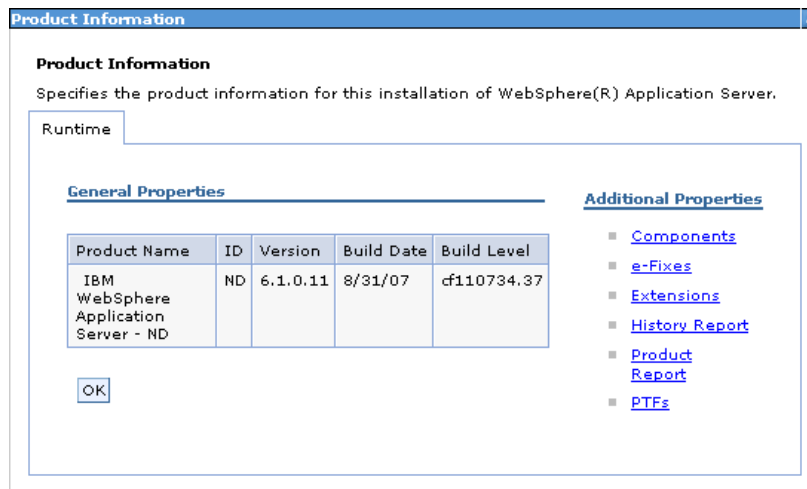


Figure 18-47. Verify current version levels and applied updates

WA5711.0

Notes:

The **versionInfo** command generates a report from data extracted from XML files in the `properties/version` folder. The report includes a list of changed components and installed or uninstalled maintenance packages.

The command syntax is:

```
versionInfo [ -format text | html ] [ -file file_name ] [ -long ] [
-maintenancePackages ] [ -maintenancePackageDetail ] [ -components ] [
-componentDetail ]
```

The **historyInfo** command generates a report from data extracted from XML files in the `properties/version` folder and the `properties/version/history` folder. The report includes a list of changed components and a history of installed or uninstalled maintenance packages.

The command syntax is:

```
historyInfo [ -format text | html ] [ -file file_name ]
[ -maintenancePackageID ID_of_maintenance_package ] [ -component
component_name ]
```


Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Installing maintenance packages (1 of 2)

- Download, unpack, and install the Update Installer
- Download the most current version of the update:
 - Interim fix
 - A single published emergency fix that resolves one or more product defects
 - Fix pack
 - Cumulative package of fixes, such as Fix Pack 7 (6.1.0.7) , regression tested
 - Refresh pack
 - Cumulative package that includes minor new features and fixes
- Copy the update to the
`<WAS_HOME>\UpdateInstaller\maintenance` directory
- Run the backupConfig command
- Run versioninfo command
- Use the Update Installer to install the interim fix, fix pack, or refresh pack
- Run versioninfo command again

Figure 18-48. Installing maintenance packages (1 of 2)

WA5711.0

Notes:

Fix packs uninstall all interim fixes applied to the release since the last fix pack was installed. Therefore, it is necessary to check the list of delivered fixes to determine if an interim fix needs to be reinstalled.

Refresh packs uninstall all fix packs and interim fixes previously applied to the release. So, it is necessary to check the list of delivered fixes to determine if an interim fix must be reinstalled.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Installing maintenance packages (2 of 2)

- Check log files in
`<WAS_HOME>\UpdateInstaller\maintenance\logs\install`
 - log.txt
 - installconfig.log
 - trace.txt
- Investigate any INSTCONFFAILED or INSTCONFPARTIALSUCCESS messages
- Try to use Update Installer to uninstall the update
- Run **restoreConfig** command if necessary

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting migration problems

After completing this topic, you should be able to:

- Identify migration tools
- Locate the relevant migration log files
- Identify log messages associated with migration problems

Figure 18-50. Troubleshooting migration problems

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Migrating an earlier version of WebSphere version 6.1.

- The following can be migrated from version 5.x or 6.0.x to version 6.1
 - Standalone application server
 - Deployment manager
- Using the Migration wizard
 - `<WAS_ROOT>\bin\migration.bat`
- Use the **WASPreUpgrade** command
 - `<WAS_ROOT>\bin\WASPreUpgrade.bat/sh backupDirectory currentWebSphereDirectory [-traceString trace_spec [-traceFile file_name]]`
- Use the **WASPostUpgrade** command
 - `<WAS_ROOT>\bin\WASPostUpgrade backupDirectory [-profileName profile_name] [-scriptCompatibility true | false] [-portBlock port_starting_number]`
 - more

Figure 18-51. Migrating an earlier version of WebSphere version 6.1.

WA5711.0

Notes:

Migrating a WebSphere Application Server Version 5.x or 6.0.x deployment manager to a Version 6.1 deployment manager moves all of the Version 5.x or 6.0.x managed nodes to become Version 5.x or 6.0.x managed nodes in the Version 6.1 cell. If you are migrating a Version 5.x or 6.0.x managed node that is part of a Version 6.1 cell to a Version 6.1 managed node, do not federate the managed node when you create it with the Profile Management tool, but do be sure that the Version 5.x or 6.0.x and Version 6.1 node names match. Migrating the Version 5.x or 6.0.x managed node to the Version 6.1 managed node also federates the node.

```
WASPostUpgrade backupDirectory [-profileName profile_name]
[-scriptCompatibility true | false] [-portBlock port_starting_number]
[-backupConfig true | false] [-replacePorts true | false] [-includeApps true
| false | script] [-keepAppDirectory true | false] [-keepDmgrEnabled true |
false] [-appInstallDirectory user_specified_directory] [-traceString
trace_spec [-traceFile file_name]]
```


Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Migration wizard

- Launch the Migration wizard from the command line:
<WAS_ROOT>\bin\migration.bat/sh
- From Windows: **Start -> IBM WebSphere -> Application Server Network Deployment V6.1 -> Migration wizard**

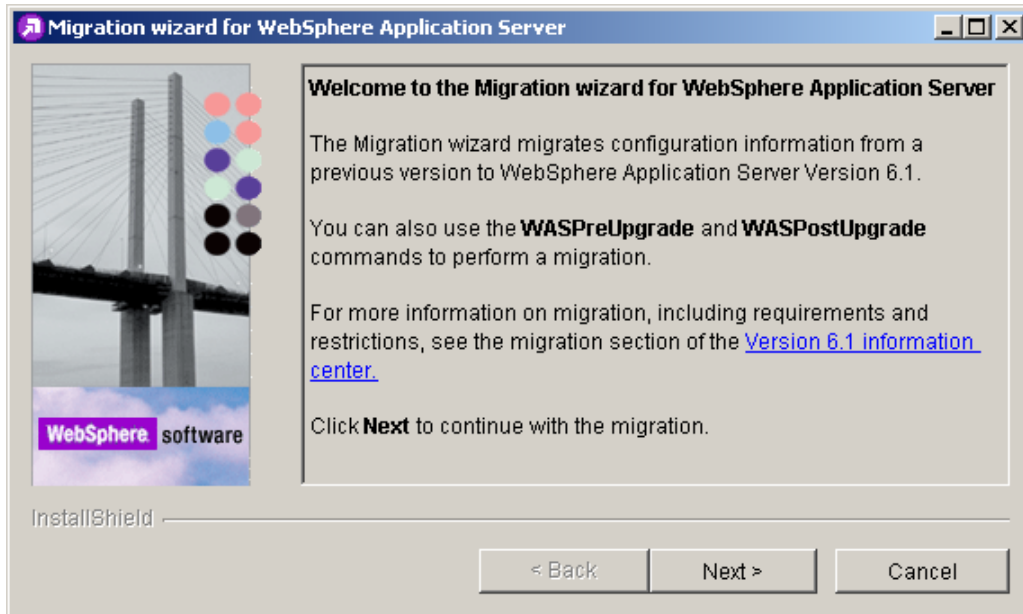


Figure 18-52. Migration wizard

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting migration (1 of 2)

- If you encounter a problem migrating from a previous version of WebSphere Application Server to Version 6.1, check the log files
- Look for the log files, and browse them for symptoms
 - <migration_backup_dir>/WASPreUpgrade.time_stamp.log
 - <profile_root>/logs/WASPostUpgrade.time_stamp.log
 - <app_server_root>/logs/clientupgrade.time_stamp.log

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Troubleshooting migration (2 of 2)

- Look for a messages similar to:
 - MIGR0259I: The migration has successfully completed.

 - MIGR0271W: The migration completed with warnings. in the migration_backup_dir/WASPreUpgrade.time_stamp.log, profile_root/logs/WASPostUpgrade.time_stamp.log, or app_server_root/logs/clientupgrade.time_stamp.log.

 - MIGR0286E: The migration failed to complete.

- If the migration is not successful attempt to correct any problems based on the error messages that appear in the log files

- Refer to the Information Center for documentation on Migration Troubleshooting
 - http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tmig_troubleshoot.html

Figure 18-54. Troubleshooting migration (2 of 2)

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Before contacting IBM Support

- Check that you have the prerequisites for:
 - Hardware
 - Software

- Use WebSphere Information Center or the following link for prerequisites:
 - <http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html>

- Determine what version and fix pack level is currently installed
 - Run the versionInfo command from `<WAS_install_root>/bin`

- Read the MustGather documentation for installation problems
 - http://www.ibm.com/support/docview.wss?rs=180&context=SSCVS24&q1=MustGatherDocument&uid=swg21255887&loc=en_US&cs=utf-8&lang=en

Figure 18-55. Before contacting IBM support

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information — The students will use the versionInfo command in the exercise.

Transition statement —

Checkpoint

1. List 4 things that can go wrong during the installation process.
2. What are some reasons profile creation will fail?
3. When should the installver tool be run?
4. What tool should be used to apply an interim fix or fix pack?

Figure 18-56. Checkpoint

WA5711.0

Notes:

Write down your answers here:

1.
 - a.
 - b.
 - c.
 - d.
2.
 - a.
 - b.
 - c.
- 3.
- 4.

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Checkpoint solutions

1. List 4 things that can go wrong during the installation process.
 - **Launchpad or installation wizard fails to start**
 - **The installation process fails**
 - **Profile creation fails**
 - **The IVT fails to start the server or fails to validate one or more components**
 - **The Installation Verification Utility reports missing files or changed files**

2. What are some reasons profile creation will fail?
 - **A profile creation will fail for the following reasons: Long directory paths, file permissions, host name, conflicting ports**

3. When should the installver tool be run?
 - **The installver tool should be run after the initial installation and then after subsequent fix pack or refresh pack updates**

4. What tool should be used to apply an interim fix or fix pack?
 - **Use the Update Installer to apply fixes, fix packs, and refresh packs**

Figure 18-57. Checkpoint solutions

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit summary

Having completed this unit, you should be able to:

- Follow installation checklist to detect installation problems
- Locate and examine relevant installation logs
- Use the Installation Verification Tool (IVT)
- Use the Installation Verification Utility (IVU)
- Recover from failed installation—uninstall and clean machine
- Run install command with different options for logging, tracing and silent install
- Troubleshoot maintenance update installation
- Troubleshoot migration problems

Figure 18-58. Unit summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Exercise

- Exercise 13: Troubleshooting installation problems

Figure 18-59. Exercise

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Unit 19. Course summary

Estimated time

00:10

What this unit is about

This unit summarizes the course.

What you should be able to do

After completing this unit, you should be able to:

- Use problem determination tools, techniques, and analyses to troubleshoot problems
- Identify common problems
- Use IBM Support Assistant
- Communicate with IBM Support teams effectively.

How you will check your progress

Accountability:

- Checkpoint
- Machine exercises

References

WebSphere: www.ibm.com/websphere

WebSphere Support page:

<http://www.ibm.com/software/webservers/appserv/was/support/>

Redbooks: <http://www.ibm.com/redbooks>

SG24-6798-00 – *WebSphere Application Server V6 Problem Determination*

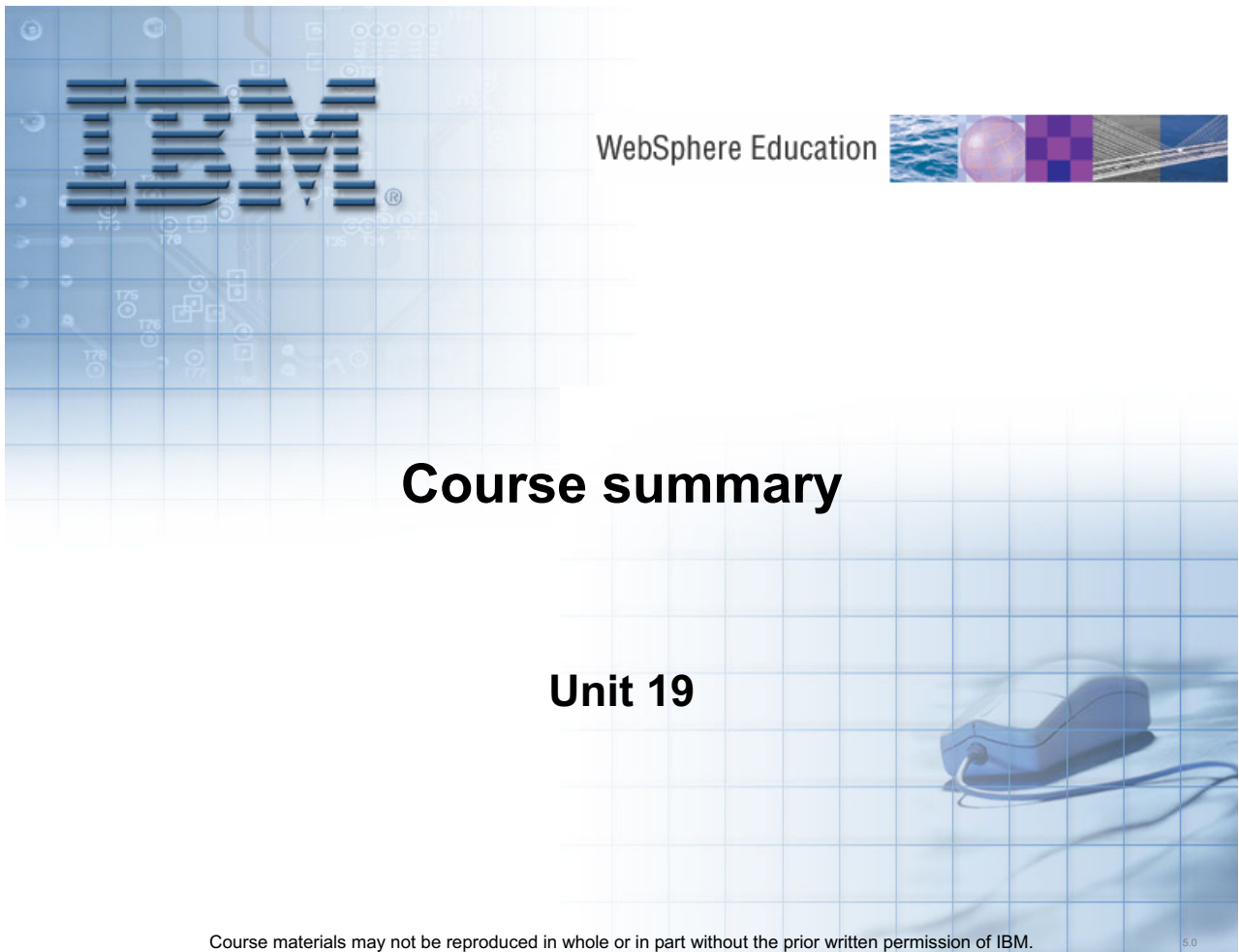


Figure 19-1. Course summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Course summary

Having completed this course, you should be able to:

- Use problem determination methodology, tools, techniques to troubleshoot problems
- Identify common problems
- Use IBM Support Assistant
- Communicate effectively with IBM support teams

Figure 19-2. Course summary

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Resources

- Web pages
 - WebSphere: www.ibm.com/websphere
 - WebSphere Support page:
<http://www.ibm.com/software/webservers/appserv/was/support/>
- Redbooks: www.ibm.com/redbooks
 - SG24-6798-00 *WebSphere Application Server V6 Problem Determination*
 - SG24-6316-01 *IBM WebSphere Application Server V6.1 Security Handbook*
- Redpapers: www.ibm.com/redbooks
 - REDP-4330-00 *WebSphere Application Server V6.1: JMS Problem Determination*
 - REDP-4306-00 *WebSphere Application Server V6.1: Web Services Problem Determination*
 - REDP-4307-00 *WebSphere Application Server V6.1: Classloader Problem Determination*
 - Several others
- News groups (news.software.ibm.com)
 - ibm.software.websphere.application-server

Figure 19-3. Resources

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Resources (continued)

- New to SOA and Web services
 - New to SOA: <http://www.ibm.com/developerworks/webservices/newto/>
 - SOA Governance:
http://www.ibm.com/software/solutions/soa/entrypoints/advancing_soa_governance.html
- Specifications
 - WSDL: <http://www.w3.org/TR/wsdl>

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Articles and white papers

- WebSphere Application Server V6 advanced security hardening - Part 1
 - http://www.ibm.com/developerworks/websphere/techjournal//0512_botzum/0512_botzum1.html
- WebSphere Application Server V6 advanced security hardening - Part 2
 - http://www.ibm.com/developerworks/websphere/techjournal//0512_botzum/0512_botzum2.html
- IBM WebSphere Developer Technical Journal: Using the Java Secure Socket Extension in WebSphere Application Server
 - http://www.ibm.com/developerworks/websphere/techjournal/0502_benantar/0502_benantar.html
- IBM JVM Diagnosis documentation:
 - <http://www.ibm.com/developerworks/java/jdk/diagnosis/>
- IBM JRE and JDK forum:
 - http://www.ibm.com/developerworks/forums/dw_forum.jsp?forum=367&start=0&thRange=15&cat=10
- Memory leak detection and analysis in WebSphere Application Server
 - http://www.ibm.com/developerworks/websphere/library/techarticles/0606_poddar/0606_poddar.html
- Garbage collection policy and tuning article on developerWorks
 - <http://www.ibm.com/developerworks/java/library/j-ibmjava3/>

Figure 19-5. Articles and white papers

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Articles and white papers (continued)

- Interpreting a WebSphere Application Server trace file
 - http://www.ibm.com/developerworks/websphere/techjournal/0704_supauth/0704_supauth.html
- Features and tools for practical troubleshooting
 - http://www.ibm.com/developerworks/websphere/techjournal/0702_supauth/0702_supauth.html
- Build a framework for problem determination triage
 - <http://www.ibm.com/developerworks/autonomic/library/ac-pdtrriage1/>
- 12 ways you can prepare for effective production troubleshooting
 - http://www.ibm.com/developerworks/websphere/techjournal/0708_supauth/0708_supauth.html

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Other WebSphere courses

- WebSphere Education:
<http://ibm.com/developerworks/websphere/education/enablement/>
 - WA361: IBM WebSphere Application Server V6.1 Administration on Windows
 - SW222: IBM WebSphere V6 Multiplatform Advanced Administration and Configuration
 - SW227: IBM WebSphere Application Server V6 Advanced Administration
 - SW242: IBM WebSphere Performance Tools and Methodology Workshop
 - SW246: IBM WebSphere Application Server V6.0 Administration
 - SW246: IBM WebSphere Application Server V5.1 Administration
 - SW248: Transition to IBM WebSphere Application Server V6 for Administrators
 - SW248: Transition to IBM WebSphere Application Server V5.1 for Administrators
 - SW249: IBM WebSphere Application Server V5.1 Administration for Linux
 - SW253: IBM WebSphere Application Server Express V6 for Administrators
 - SW270: IBM WebSphere Application Server for iSeries V5.1 Basic Administration
 - SW271: IBM WebSphere Application Server for iSeries V5.1 Advanced Administration
 - SW279: IBM WebSphere Application Server - Express on iSeries Administration
 - SW341: From J2EE Applications to Business Processes: IBM WebSphere V6 Administration
 - SW512: IBM WebSphere Application Server V6.0 Administration on iSeries
 - SW516: IBM WebSphere Application Server Express V6.0 Administration on iSeries
 - SW517: IBM WebSphere Application Server V6 Performance Tuning for iSeries
 - SW570: IBM WebSphere Application Server V6.0 Problem Determination
 - WF881: IBM WebSphere V6 Performance Monitoring and Tuning for Administrators
 - WF681: WebSphere Automation/Advanced Scripting

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

Evaluations

- WebSphere Education is interested in your impressions of this class.
- You can provide feedback using the course evaluation forms.



Figure 19-8. Evaluations

WA5711.0

Notes:

Instructor notes:

Purpose —

Details —

Additional information —

Transition statement —

