

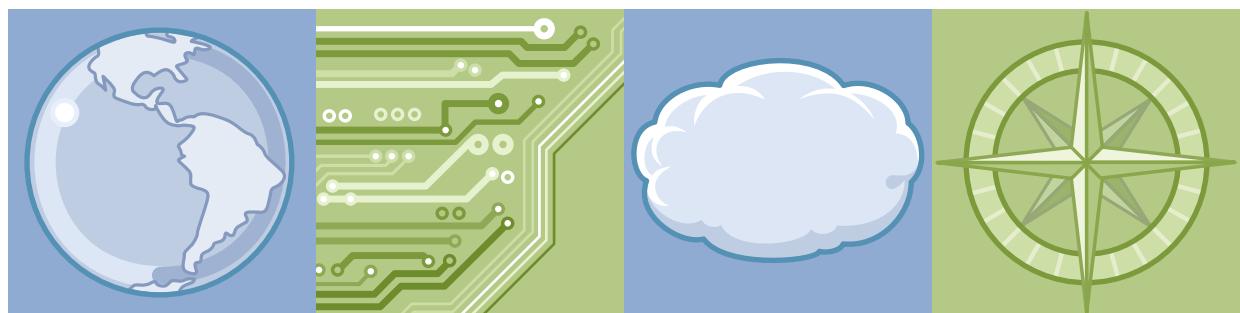


IBM Training

Student Notebook

WebSphere Application Server V8.5.5 Problem Determination

Course code WA591 ERC 3.0



IBM Cloud

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AFS™	AIX 6™	AIX®
Approach®	DB™	DB2®
developerWorks®	HACMP™	IMS™
MVS™	PartnerWorld®	RACF®
Rational®	RDN®	Redbooks®
Redpaper™	Tivoli®	WebSphere®
WPM®	z/OS®	

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

June 2017 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.

Contents

Trademarks	xxi
Course description	xxv
Agenda	xxvii
Unit 1. Overview of WebSphere Application Server systems and components	1-1
Unit objectives	1-2
Topics	1-3
1.1. Components and services.....	1-5
Components and services	1-6
WebSphere Application Server architecture	1-7
Web container	1-9
Enterprise JavaBeans (EJB) container	1-11
Java EE Connector Architecture service (JCA)	1-13
Transaction service	1-15
Data replication service	1-16
Name service	1-18
Naming topology	1-19
Object Request Broker service	1-21
WebSphere default messaging	1-23
Security service	1-24
Security domains	1-25
1.2. Network Deployment components.....	1-27
Network Deployment components	1-28
Network Deployment concepts	1-29
Runtime flow	1-30
Administration flow	1-31
File synchronization and file transfer	1-32
Clustered application servers	1-34
Web servers	1-35
Web server plug-ins	1-36
Web servers: Managed nodes	1-37
Web servers: Unmanaged nodes	1-38
High availability (HA) manager	1-39
Flexible management	1-41
Edge Components	1-43
Liberty profile	1-44
Liberty profile: Composable runtime	1-45
Intelligent Management (1 of 2)	1-46
Intelligent Management (2 of 2)	1-47
1.3. Client request flow.....	1-49
Client request flow	1-50
Typical client request application flow	1-51
Are all components in the application flow accessible?	1-53
Unit summary	1-54
Checkpoint questions (1 of 2)	1-55
Checkpoint questions (2 of 2)	1-56

Checkpoint answers	1-57
Unit 2. Using the IBM Support Assistant Team Server 5.0	2-1
Unit objectives	2-2
Topics	2-3
2.1. IBM Support Assistant overview	2-5
IBM Support Assistant overview	2-6
What is the IBM Support Assistant Team Server 5.0?	2-7
Getting started with IBM Support Assistant 5	2-8
Banner: Features	2-9
2.2. IBM Support Assistant Team Server administration	2-11
IBM Support Assistant Team Server administration	2-12
Administration: Application Settings	2-13
Administration: Tools Administration	2-14
Administration: Tool installation state indicators	2-15
Administration: Tool catalog details pane	2-16
Administration: Tool tags	2-17
Administration: Tool installation	2-18
2.3. Case management	2-19
Case management	2-20
Case management	2-21
Files tab: Add files to a case	2-22
Files tab: Operating on case files	2-23
2.4. Problem determination tools	2-25
Problem determination tools	2-26
Problem determination tools: Overview	2-27
Editor's choice and recommended tools	2-28
Problem determination tools: Launching tools	2-29
Problem determination tools: Report generators	2-30
Problem determination tools: Web tools	2-32
Problem determination tools: Desktop tools	2-33
Tools tab: Overview	2-34
Tools tab: List of installed tools	2-35
Tools tab: Details pane	2-36
Reports tab	2-37
2.5. Automated analysis	2-39
Automated analysis	2-40
Automated analysis overview	2-41
Automated analysis: Perform a scan	2-42
Automated analysis: Filtered Files tab (Flat view)	2-43
Automated analysis: Overview tab	2-44
Automated analysis: Symptoms tab	2-45
Automated analysis: Knowledge tab	2-46
2.6. Installing IBM Support Assistant 5	2-47
Installing IBM Support Assistant 5	2-48
Installing IBM Support Assistant 5: Overview	2-49
Installation Manager: IBM ID credentials	2-50
Installation Manager: Install packages	2-51
Installation Manager: License agreements	2-52
Installation Manager: Create package group	2-53
Installation Manager: Select features	2-54

Installation Manager: Confirm installation packages	2-55
Unit summary	2-56
Checkpoint questions	2-57
Checkpoint answers	2-58
Exercise 1	2-59
Exercise objectives	2-60
Unit 3. Problem determination methods	3-1
Unit objectives	3-2
Topics	3-3
3.1. Preparation: Before problems occur	3-5
Preparation: Before problems occur	3-6
Key steps for problem determination	3-7
Prepare for effective production troubleshooting	3-9
Monitoring and problem detection	3-12
System architecture or topology diagram	3-14
Example: Topology diagram	3-15
Establish baselines	3-16
Diagnostic data collection plan (1 of 2)	3-18
Diagnostic data collection plan (2 of 2)	3-20
Maintenance plans	3-22
3.2. Organize the investigation	3-23
Organize the investigation	3-24
Key steps for problem determination	3-25
Characterize the problem (1 of 2)	3-26
Characterize the problem (2 of 2)	3-27
Example: Table of issues and symptoms	3-28
High-level timeline	3-29
3.3. Relief options	3-31
Relief options	3-32
Key steps for problem determination	3-33
Relief or recovery plan	3-34
Relief options	3-35
3.4. Phase 1: Initial investigation	3-37
Phase 1: Initial investigation	3-38
Key steps for problem determination	3-39
“Solve a problem” flow: The big picture	3-40
Phase 1: Initial investigation (1 of 2)	3-41
Phase 1: Initial investigation (2 of 2)	3-42
3.5. Phase 2: In-depth investigation	3-43
Phase 2: In-depth investigation	3-44
Key steps for problem determination	3-45
“Solve a problem” flow: The big picture	3-46
Phase 2: In-depth investigation	3-47
Problem determination techniques (1 of 2)	3-48
Problem determination techniques (2 of 2)	3-49
Unit summary	3-50
Checkpoint questions	3-51
Checkpoint answers	3-52

Unit 4. Gathering diagnostic data	4-1
Unit objectives	4-2
Topics	4-3
4.1. Resources for a problem investigation	4-5
Resources for a problem investigation	4-6
WebSphere knowledge base	4-7
IBM Support Portal	4-8
WebSphere Support Portal: Troubleshooting	4-9
Information center: Troubleshooting and support	4-10
MustGather example (1 of 3)	4-11
MustGather example (2 of 3)	4-12
MustGather example (3 of 3)	4-13
Using the IBM Support Assistant Data Collector (1 of 2)	4-14
Using the IBM Support Assistant Data Collector (2 of 2)	4-15
4.2. Data gathering tools.	4-17
Data gathering tools	4-18
Types of tools and where to find them (1 of 4)	4-19
Types of tools and where to find them (2 of 4)	4-21
Types of tools and where to find them (3 of 4)	4-22
Types of tools and where to find them (4 of 4)	4-23
4.3. Logs and tracing: Basic mode	4-25
Logs and tracing: Basic mode	4-26
Basic mode logs and tracing	4-27
Log files and locations	4-28
Configuring JVM logs	4-30
Viewing runtime messages in the console	4-31
HTTP plug-in logs and tracing	4-32
Embedded HTTP server logs	4-33
Diagnostic traces	4-34
Enabling diagnostic trace	4-36
Diagnostic trace dump and runtime	4-38
Setting the log detail level	4-39
Enabling and disabling sensitive log and trace guard	4-41
Enabling trace by using Jython scripts (1 of 2)	4-43
Enabling trace by using Jython scripts (2 of 2)	4-44
JVM log and trace log headers	4-45
Default trace format	4-46
Reading a log or trace file (1 of 2)	4-48
Reading a log or trace file (2 of 2)	4-50
4.4. High Performance Extensible Logging (HPEL)	4-51
High Performance Extensible Logging (HPEL) (1 of 3)	4-52
High Performance Extensible Logging (HPEL) (2 of 3)	4-53
High Performance Extensible Logging (HPEL) (3 of 3)	4-55
HPEL logging and tracing configuration	4-56
HPEL log and trace storage	4-57
Log files and locations for HPEL	4-58
Contents of the logs directory	4-60
Configure HPEL logging	4-61
HPEL Log Viewer	4-62
HPEL Log Viewer: Filtering options	4-64
HPEL LogViewer: Command-line tool (1 of 2)	4-66

HPEL LogViewer: Command-line tool (2 of 2)	4-68
What is Cross Component Trace (XCT)?	4-69
Administering XCT	4-70
XCT request IDs	4-71
Use XCT request ID information to track requests	4-72
4.5. Gathering JVM-related data	4-75
Gathering JVM-related data	4-76
Enable verbose garbage collection	4-77
Java dumps and cores	4-78
Dumping the JNDI name space	4-80
Dumping the JNDI name space example	4-81
Unit summary	4-82
Checkpoint questions	4-83
Checkpoint answers	4-84
Exercise 2	4-85
Exercise objectives	4-86
Unit 5. Introduction to JVM-related problems	5-1
Unit objectives	5-3
Topics	5-4
5.1. JVM introduction	5-5
JVM introduction	5-6
Java virtual machine (JVM) features	5-7
Just-in-time compiler (JIT) basics	5-8
Ahead-Of-Time (AOT) compiler basics	5-9
JVM version (1 of 2)	5-10
JVM version (2 of 2)	5-12
JVM overview	5-13
JVM memory segments	5-15
Understanding garbage collection (GC)	5-17
IBM JDK GC process	5-18
Garbage collection policies	5-20
Generational concurrent GC (gencon)	5-22
Generational concurrent GC: Scavenge (1 of 2)	5-23
Generational concurrent GC: Scavenge (2 of 2)	5-24
Generational concurrent GC: Self-adjusting	5-25
Using IBM Java 7	5-26
5.2. Introduction to JVM problem determination	5-29
Introduction to JVM problem determination	5-30
JVM problem determination: Symptom analysis	5-31
JVM problem determination: Data to collect	5-32
Javacore overview	5-33
Javacore file location and naming	5-34
Javacore file subcomponents	5-36
Javacore example (JDK v6)	5-38
Verbose garbage collection (GC)	5-39
5.3. JVM tuning	5-41
JVM tuning	5-42
Tuning method	5-43
Maximum heap size	5-44
Theoretical and advised maximum heap sizes (1 of 2)	5-45

Theoretical and advised maximum heap sizes (2 of 2)	5-46
Tuning considerations: Heap	5-47
The “correct” Java heap size (1 of 2)	5-48
The “correct” Java heap size (2 of 2)	5-49
Fixed heap sizes versus variable heap sizes	5-50
Tuning considerations: Garbage collection	5-51
Tuning considerations: GC policy	5-53
Tuning considerations: Native heap	5-54
JVM tool overview	5-55
Unit summary	5-56
Checkpoint questions	5-57
Checkpoint answers	5-59
Exercise 3	5-60
Exercise objectives	5-61
 Unit 6. How to troubleshoot hangs	 6-1
Unit objectives	6-2
Topics	6-3
6.1. Application server hangs	6-5
Application server hangs	6-6
Application server hangs defined	6-7
Thread hangs	6-8
Thread hang examples in application code	6-9
WebSphere process hangs detection steps	6-10
How to manually trigger a thread dump	6-12
Thread dump analysis	6-14
Javacore hang indicators	6-15
Thread state values	6-17
Java 7: More details about monitor usage	6-18
Javacore hang symptoms (1 of 2)	6-20
Javacore hang symptoms (2 of 2)	6-21
Hang detection tools	6-22
6.2. Hung thread detection	6-23
Hung thread detection	6-24
WebSphere hung thread detection	6-25
Hung thread detection internals	6-26
Hung thread detection notification	6-27
Hung thread detection false alarms	6-29
Hung thread detection configuration	6-30
6.3. IBM Thread and Monitor Dump Analyzer	6-33
IBM Thread and Monitor Dump Analyzer	6-34
What is Thread and Monitor Dump Analyzer (TMDA)?	6-35
TMDA: Open a thread dump	6-36
Thread detail: Thread status analysis	6-38
Thread detail: Thread method analysis	6-40
Thread detail: Thread aggregation analysis	6-41
Thread detail: Memory segment analysis	6-42
Multiple dump comparative analysis	6-43
Thread analysis: Deadlocked thread details	6-44
Thread analysis: Monitor details	6-45
IBM Whole-system Analysis of Idle Time (WAIT)	6-46

Example WAIT report	6-47
Unit summary	6-49
Checkpoint	6-50
Checkpoint answers	6-51
Exercise 4	6-52
Exercise objectives	6-53
Unit 7. How to troubleshoot crashes	7-1
Unit objectives	7-2
Topics	7-3
7.1. Troubleshooting a crash	7-5
Troubleshooting a crash	7-6
JVM process crash defined	7-7
Crash problem determination: Data to collect	7-8
Make sure that a full core dump is produced	7-10
Diagnostics Collector	7-11
UNIX operating system common signals	7-13
Windows operating system common signals	7-15
Javacore subcomponents helpful for crash debugging	7-16
Javacore example that shows crash symptoms	7-18
Javacore fault module	7-19
Javacore current thread details (JDK 1.4.2)	7-20
Crashes during JIT compilation	7-21
Steps if the cause of the crash is not identified	7-22
7.2. Tools for troubleshooting crashes	7-23
Tools for troubleshooting crashes	7-24
What is DTFJ?	7-25
Components of the DTFJ family	7-26
An example of using the DTFJ components	7-27
Where is DTFJ supported?	7-28
What is the Interactive Diagnostic Data Explorer?	7-29
Using Interactive Diagnostic Data Explorer (IDDE)	7-30
IDDE features	7-31
IDDE: jvm.DumpReport contents	7-32
IDDE: dump report	7-33
IDDE: Help menu for DTFJ commands	7-34
IDDE: Deadlocked threads detection	7-35
IDDE: JIT compiled methods	7-36
Unit summary	7-37
Checkpoint questions	7-38
Checkpoint answers	7-39
Exercise 5	7-40
Exercise objectives	7-41
Unit 8. Introduction to WebSphere out-of-memory problems	8-1
Unit objectives	8-2
Topics	8-3
8.1. OutOfMemory error overview	8-5
OutOfMemory error overview	8-6
What is a java.lang.OutOfMemoryError?	8-7
JVM heap is too small	8-8

Memory fragmentation	8-9
Symptoms of memory leak in the Java code	8-10
Classloader memory leaks	8-11
Not enough native memory	8-13
Java process restrictions	8-14
The native heap	8-15
Using Tivoli Performance Viewer to anticipate an OutOfMemoryError	8-16
Administrative console PMI settings	8-17
Administrative console Tivoli Performance Viewer monitoring	8-18
Administrative console Tivoli Performance Viewer graph	8-19
Tivoli Performance Viewer graph: Heap expansion	8-20
Tivoli Performance Viewer usage	8-21
OutOfMemory indicators in the javacore file	8-23
How to obtain a verboseGC log	8-24
Default behavior for OutOfMemory exceptions	8-25
System dump files	8-26
Obtaining system core dumps	8-27
Java heap dumps	8-28
Heap dumps versus system core dumps	8-29
Obtaining Java heap dumps (1 of 3)	8-30
Obtaining Java heap dumps (2 of 3)	8-31
Obtaining Java heap dumps (3 of 3)	8-32
Use wsadmin to trigger a heap dump	8-33
Java heap dumps by using dump agents	8-34
Generic arguments that use Xdump:heap	8-35
Generic arguments that use Xdump:system	8-36
Automatic heap dump generation with IBM JDK V6	8-37
8.2. Interpret verbose GC output	8-39
Interpret verbose GC output	8-40
OutOfMemory: Interpret verboseGC output	8-41
JVM messages that show dump events	8-42
Verbose GC output for optthruput policy	8-43
Java 6.26 GC output: optthruput policy (1 of 3)	8-44
Java 6.26 GC output: optthruput policy (2 of 3)	8-46
Java 6.26 GC output: optthruput policy (3 of 3)	8-47
Verbose GC output for gencon policy	8-48
Java 6.26 GC output: gencon policy (1 of 3)	8-49
Java 6.26 GC output: gencon policy (2 of 3)	8-50
Java 6.26 GC output: gencon policy (3 of 3)	8-51
Verbose GC output for gencon failures	8-52
IBM Monitoring and Diagnostic Tools for Java – Garbage Collection and Memory Visualizer (GCMV)	8-53
Garbage Collection and Memory Visualizer overview	8-54
GCMV usage scenarios	8-55
Plotting verbose GC data with the GCMV	8-56
Types of graphs	8-58
Garbage collection trigger graphs	8-59
Heap usage and occupancy recommendation (1 of 2)	8-60
Heap usage and occupancy recommendation (2 of 2)	8-61
Comparing verbose GC between different JVMs (1 of 2)	8-62
Comparing verbose GC between different JVMs (2 of 2)	8-63

IBM Pattern Modeling and Analysis Tool (PMAT)	8-64
PMAT: Chart view	8-65
8.3. Analyze Java heap dumps and system core dumps	8-67
Analyze Java heap dumps and system core dumps	8-68
OutOfMemory: Interpret heap dumps	8-69
How to analyze a Java heap dump	8-70
IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer Version 1.1	8-71
Memory Analyzer overview	8-72
Memory Analyzer leak suspects: default report	8-74
Denominator tree view	8-75
Path to GC roots	8-76
Thread stack view	8-77
IBM Extensions for Memory Analyzer	8-78
IBM Extensions for Memory Analyzer	8-80
HeapAnalyzer	8-82
HeapAnalyzer summary	8-83
HeapAnalyzer tree	8-84
Unit summary	8-85
Checkpoint questions	8-86
Checkpoint answers	8-87
Exercise 6	8-88
Exercise objectives	8-89
Unit 9. Introduction to database connection problems	9-1
Unit objectives	9-2
Topics	9-3
9.1. Overview of database connection concepts	9-5
Overview of database connection concepts	9-6
JDBC providers	9-7
Data sources	9-8
Enhanced EAR	9-9
The role of the JDBC provider	9-10
JDBC provider configuration	9-11
The role of the data source	9-13
Data source screen (1 of 3)	9-15
Data source screen (2 of 3)	9-16
Data source screen (3 of 3)	9-18
9.2. Common database connection problems	9-19
Common database connection problems	9-20
What can go wrong	9-21
Creating a database connection: Common problems	9-22
Using the test connection service (1 of 2)	9-24
Using the test connection service (2 of 2)	9-25
JDBC provider configuration problems: Other considerations	9-26
Data source database parameter problems: Identification	9-27
Data source database parameter problems: Diagnosis and resolution	9-28
Data source database authentication problems: Identification	9-29
Data source database authentication problems: Diagnosis and resolution	9-30
Database deadlock problems	9-31
Database lock contention and rollback problems	9-32
Overview of the DB2 Universal Driver trace facility	9-33

Enabling the DB2 Universal Driver trace facility	9-34
DB2 Universal Driver trace facility output example	9-35
Enabling Oracle JDBC driver tracing	9-36
Unit summary	9-37
Checkpoint questions	9-38
Checkpoint answers	9-39
Exercise 7	9-40
Exercise objectives	9-41
Unit 10. Tuning and connection pool management problems	10-1
Unit objectives	10-2
Topics	10-3
10.1. Connection pooling	10-5
Connection pooling	10-6
What is connection pooling?	10-7
JCA connection pooling architecture	10-8
Types of connection pools in WebSphere	10-10
Connection lifecycle	10-11
Shareable versus unshareable connections (1 of 2)	10-13
Shareable versus unshareable connections (2 of 2)	10-15
Detecting connection management-related problems	10-16
Typical connection pool runtime problem symptoms	10-17
Connection pooling problem determination path	10-18
10.2. Troubleshooting configuration problems	10-19
Troubleshooting configuration problems	10-20
Troubleshooting connection pool configuration in the problem determination path	10-21
The need for tuning connection pools	10-22
Key connection pool parameters	10-23
Connection pool parameters in the administrative console	10-24
Task for tuning a connection pool	10-26
Monitoring the connection pool by using Tivoli Performance Viewer	10-27
Tivoli Performance Viewer connection pool monitoring example	10-29
Generating tuning advice by using TPV Performance Advisor	10-30
TPV Performance Advisor example	10-32
Tuning the connection pool	10-34
Suggestions for tuning connection pools	10-35
10.3. Troubleshooting connection leaks	10-37
Troubleshooting connection leaks	10-38
Troubleshooting connection leaks in the problem determination path	10-39
Connection leaks	10-40
Common causes of connection leaks	10-41
Connection leak diagnosis tasks	10-42
Connection leak trace facility	10-43
Configuring the connection leak trace facility	10-45
What to look for in the trace	10-47
Troubleshooting ConnectionWaitTimeoutException errors	10-48
Output from “showPoolContents”	10-49
10.4. Troubleshooting stale connections	10-51
Troubleshooting stale connections	10-52
Troubleshooting stale connections in the problem determination path	10-53
Stale connections	10-54

Recovering from a stale connection	10-55
JCA lifecycle management	10-56
Other stale connection troubleshooting tasks	10-57
Connection pool troubleshooting tools (1 of 2)	10-58
Connection pool troubleshooting tools (2 of 2)	10-59
Unit summary	10-61
Checkpoint questions	10-62
Checkpoint answers	10-63
Exercise 8	10-64
Exercise objectives	10-65
Unit 11. WebSphere security configuration problems	11-1
Unit objectives	11-2
Topics	11-3
11.1. Review of security components and security flows	11-5
Review of security components and security flows	11-6
Security components overview	11-7
Authentication flows	11-9
Security flows: web browser communication	11-11
Security flows: Java client communication	11-12
Authorization flows	11-13
Federated repositories	11-14
Security domains	11-15
11.2. Common problems and troubleshooting methods	11-17
Common problems and troubleshooting methods	11-18
What can go wrong? The short list (1 of 2)	11-19
What can go wrong? The short list (2 of 2)	11-20
Approach to troubleshooting security-related issues	11-21
Normal security initialization messages	11-22
Authentication or authorization problem	11-23
Problems that are related to Secure Sockets Layer (SSL)	11-24
Stack trace in the system log file	11-25
Example: SystemOut.log stack trace	11-26
Example: SystemOut.log exceptions	11-27
Tracing security components	11-28
MustGather tracing requirements	11-29
Result from security components trace	11-30
11.3. SSL problems	11-31
SSL problems	11-32
SSL use in WebSphere	11-33
How does SSL work? The “handshake”	11-34
What is the key ring, keystore, and a truststore?	11-35
SSL client/server endpoint identification	11-36
SSL problems: Handshake failures (1 of 3)	11-37
SSL problems: Handshake failures (2 of 3)	11-39
SSL problems: Handshake failures (3 of 3)	11-40
Common SSL connection error message	11-41
Steps to diagnose SSL handshake issues (1 of 3)	11-43
Steps to diagnose SSL handshake issues (2 of 3)	11-45
Steps to diagnose SSL handshake issues (3 of 3)	11-46
General SSL problem determination steps	11-47

11.4. Java 2 security problems	11-49
Java 2 security problems	11-50
Java 2 security problems (1 of 2)	11-51
Java 2 security problems (2 of 2)	11-52
11.5. More tools and techniques	11-53
More tools and techniques	11-54
Administrative console security PD tools (1 of 3)	11-55
Administrative console security PD tools (2 of 3)	11-56
Administrative console security PD tools (3 of 3)	11-57
Security configuration files (1 of 3)	11-58
Security configuration files (2 of 3)	11-59
Security configuration files (3 of 3)	11-60
Tracking LTPA tokens	11-61
Disabling administrative security	11-63
Unit summary	11-64
Checkpoint questions	11-65
Checkpoint answers	11-66
Exercise 9	11-67
Exercise objectives	11-68
Unit 12. Application deployment problems	12-1
Unit objectives	12-2
Topics	12-3
12.1. Application installation problems	12-5
Application installation problems	12-6
Application installation methods: Console wizard	12-7
Application installation methods: wsadmin	12-9
Application installation methods: Monitored directory	12-10
Problem areas in application installation	12-11
Examining console and error log file messages	12-12
wsadmin command-line exceptions	12-14
Typical application installation errors	12-15
Typical application startup errors	12-17
Typical application first-run errors (1 of 2)	12-18
Typical application first-run errors (2 of 2)	12-19
Deployment troubleshooting tools (1 of 2)	12-20
Deployment troubleshooting tools (2 of 2)	12-21
Gathering trace data for deployment problems	12-23
12.2. Troubleshooting class loader problems	12-25
Troubleshooting class loader problems	12-26
Class loading concepts (1 of 2)	12-27
Class loading concepts (2 of 2)	12-29
Class loader modes and policies	12-31
Class Loader Viewer: Application topology	12-32
Class loading problems (1 of 2)	12-34
Class loading problems (2 of 2)	12-35
How to resolve a ClassNotFoundException problem	12-37
Unit summary	12-39
Checkpoint questions	12-40
Checkpoint answers	12-41

Unit 13. Server start failures	13-1
Unit objectives	13-2
Topics	13-3
13.1. Troubleshooting server start failures	13-5
Troubleshooting server start failures	13-6
What happens when a server starts	13-7
Server start messages	13-9
Server bootup process	13-10
Starting a server from the administrative console	13-11
Detecting a failed server start (1 of 3)	13-12
Detecting a failed server start (2 of 3)	13-14
Detecting a failed server start (3 of 3)	13-15
Starting a server from the console – failure	13-17
Common causes of server start failures	13-18
Determining server start issues (1 of 3)	13-19
Determining server start issues (2 of 3)	13-23
Determining server start issues (3 of 3)	13-25
Common configuration error messages	13-26
Resolving server start failures (1 of 2)	13-28
Resolving server start failures (2 of 2)	13-29
IBM Port Scanning Tool	13-30
13.2. Troubleshooting server stop failures	13-31
Troubleshooting server stop failures	13-32
Server stop failures	13-33
Other administrative console stop server options	13-34
Stop server by killing the Java process	13-35
Restarting an application server in recovery mode	13-36
Windows services: Monitoring server processes	13-38
A tool to help create Windows services	13-39
UNIX: Automatic server monitoring and restart	13-40
13.3. Node synchronization and node discovery failures	13-41
Node synchronization and node discovery failures	13-42
Node agent and deployment manager discovery process	13-43
Important addresses in the discovery process	13-44
Failure of a managed node to synchronize	13-45
13.4. Validate configuration	13-47
Validate configuration	13-48
Monitoring configuration files from the administrative console	13-49
WebSphere Configuration Problems: validation policy	13-50
Configuration problems	13-52
Unit summary	13-53
Checkpoint questions	13-54
Checkpoint answers	13-55
Exercise 10	13-56
Exercise objectives	13-57
Unit 14. Request flow and web container problems	14-1
Unit objectives	14-2
Topics	14-3
14.1. Overview of web request flow	14-5
Overview of web request flow	14-6

Web request protocol	14-7
Request flow HTTP or HTTPS	14-9
Detailed web request flow	14-11
Typical request methods: GET	14-13
Identifying a working request: Web server access log	14-14
Identifying a working request: Plug-in log (GET)	14-16
Identifying a working request: Plug-in log (POST)	14-18
Enabling IBM HTTP Server trace	14-19
Enabling plug-in trace	14-20
Enabling web container trace	14-21
Web container diagnostic provider	14-22
14.2. Troubleshooting a failing request	14-23
Troubleshooting a failing request	14-24
Preliminary questions (1 of 2)	14-25
Preliminary questions (2 of 2)	14-27
A word about HTTP response codes	14-28
Web server generates a 404 response	14-29
Web server generates a 404 response: Solution	14-30
Plug-in fails to route a request to application server	14-31
Plug-in causing a 404 response: HTTP plug-in	14-32
Application server generates a 404 response	14-34
Plug-in generating a 500 response: Solution	14-35
HTTP plug-in 500 level generated	14-36
WebSphere Application Server 500 response	14-37
Solving a 500 response	14-38
14.3. Troubleshooting HTTP session issues	14-39
Troubleshooting HTTP session issues	14-40
Troubleshooting HTTP session issues (1 of 3)	14-41
Troubleshooting HTTP session issues (2 of 3)	14-43
Troubleshooting HTTP session issues (3 of 3)	14-45
14.4. Troubleshooting dynamic cache issues	14-47
Troubleshooting dynamic cache issues	14-48
Purpose of the dynamic cache	14-49
Troubleshooting the dynamic cache service: Servlets	14-50
Cache Monitor: Overview	14-51
Cache Monitor: managing cache data	14-52
14.5. Using diagnostic providers to troubleshoot web container issues	14-53
Using diagnostic providers to troubleshoot web container issues	14-54
What is a diagnostic provider?	14-55
Diagnostic Provider utility	14-56
State dump from WebContainerDP	14-58
Configuration dump from WebContainerDP (1 of 2)	14-59
Configuration dump from WebContainerDP (2 of 2)	14-60
Unit summary	14-61
Checkpoint questions	14-62
Checkpoint answers	14-63
Exercise 11	14-64
Exercise objectives	14-65
Unit 15. Default messaging provider problem determination	15-1
Unit objectives	15-2

Topics	15-3
15.1. WebSphere default messaging concepts	15-5
WebSphere default messaging concepts	15-6
What is JMS?	15-7
Java EE administrative components used by JMS	15-8
WebSphere default messaging	15-9
Administrative components of default messaging	15-10
Runtime components of default messaging	15-12
Runtime flow for JMS messaging (1 of 4)	15-14
Runtime flow for JMS messaging (2 of 4)	15-16
Runtime flow for JMS messaging (3 of 4)	15-17
Runtime flow for JMS messaging (4 of 4)	15-19
Runtime flow for JMS messaging (more complex)	15-20
15.2. Default messaging problem determination	15-21
Default messaging problem determination	15-22
How to approach a generic problem with WebSphere default messaging	15-23
Check that the messaging engine is in started state	15-24
Messaging engine startup typical messages (1 of 2)	15-26
Messaging engine startup typical messages (2 of 2)	15-27
Check the SystemOut.log	15-28
Check the FFDCs for details	15-29
Perform phase 1 problem determination	15-30
Use MustGather document more detailed analysis	15-31
Engage IBM support	15-32
15.3. Common messaging problems	15-33
Common messaging problems	15-34
WebSphere default messaging problem solving	15-35
ME startup problems	15-36
ME startup failures: message store problems	15-37
ME startup failures: message store problems	15-38
ME startup failures: Database UUID problems	15-39
ME startup failures: Database lock problems	15-40
Application connection issues	15-42
Application connection issues: Service unavailable	15-43
Application connection issues: Unresponsive server	15-44
Application connection issues: MDB initialization	15-45
Message flow problems (1 of 4)	15-46
Message flow problems (2 of 4)	15-47
Message flow problems (3 of 4)	15-48
Message flow problems (4 of 4)	15-49
Message flow problems: Messages not consumed	15-50
Message flow problems: States (1 of 2)	15-51
Message flow problems: States (2 of 2)	15-52
Message flow problems: Lost messages	15-53
Message flow problems: Lost messages	15-54
15.4. Messaging components and trace groups	15-55
Messaging components and trace groups	15-56
Components and their trace groups (1 of 3)	15-57
Components and their trace groups (2 of 3)	15-58
Components and their trace groups (3 of 3)	15-59
Tracing for WebSphere default messaging (1 of 3)	15-60

Tracing for WebSphere default messaging (2 of 3)	15-61
Tracing for WebSphere default messaging (3 of 3)	15-62
Service Integration Bus Destination Handler (SIBDH)	15-63
Using SIBDH to send and delete a message	15-64
Viewing ME status from the administrative console	15-66
Unit summary	15-67
Checkpoint questions	15-68
Checkpoint answers	15-69
Exercise 12	15-70
Exercise objectives	15-71
Unit 16. WebSphere installation problems when using IBM Installation Manager	16-1
Unit objectives	16-2
Topics	16-3
16.1. Introduction to IBM Installation Manager	16-5
Introduction to IBM Installation Manager	16-6
Installation Manager	16-7
InstallShield Multiplatform versus Installation Manager	16-8
Installation Manager: GUI mode (1 of 2)	16-9
Installation Manager: GUI mode (2 of 2)	16-10
Installation Manager: Silent installation	16-11
Installation Manager: Command line	16-12
Installation Manager: Console mode	16-13
IBM Installation Manager repositories	16-14
IBM Packaging Utility	16-16
Important directories: Agent data location	16-17
Default agent data location	16-18
Contents of agent data	16-19
Important directories: Agent data location	16-20
Important directories: Shared directory	16-21
16.2. How to view Installation Manager logs	16-23
How to view Installation Manager logs	16-24
Installation Manager logs and history (1 of 2)	16-25
Installation Manager logs and history (2 of 2)	16-27
Logs: Systems without graphical interface	16-28
Logs: View history by using a web browser	16-29
Logs: View logs by using a web browser (1 of 2)	16-30
Logs: View logs by using a web browser (2 of 2)	16-31
View all installed products (1 of 3)	16-32
View all installed products (2 of 3)	16-33
View all installed products (3 of 3)	16-34
16.3. Installation problem determination	16-35
Installation problem determination	16-36
Installation problem determination	16-37
Installing WebSphere Application Server	16-38
What can go wrong	16-39
Installation Manager Install and startup Issues	16-40
Issue: Installation Manager installed to default location in silent mode (1 of 2)	16-41
Issue: Installation Manager installed to default location in silent mode (1 of 2)	16-42
16.4. Failure of the installation, update, and uninstall processes	16-43
Failure of the install, update, and uninstall processes	16-44

Issues with installing offerings	16-45
Issue: Installation Manager hangs during when repositories on NFS mount	16-46
Issue: IMShared is using too much disk space	16-47
Issue: Installation Manager is not able to find an installed offering	16-48
Using the Installation Manager reinstall option	16-49
Recovering from a failed or hung installation	16-50
Collecting data for install, update, and uninstall issues	16-51
16.5. Profile creation problems.....	16-53
Profile creation problems	16-54
Profile creation failure	16-55
Verify that profile creation succeeded	16-56
Profile creation steps	16-57
Data to collect if profile creation fails (1 of 3)	16-58
Data to collect if profile creation fails (2 of 3)	16-59
Data to collect if profile creation fails (3 of 3)	16-61
What to look for if profile creation fails	16-62
16.6. Centralized Installation Manager (CIM).....	16-65
Centralized Installation Manager (CIM)	16-66
Centralized Installation Manager (CIM) in the job manager	16-67
Centralized installation manager problems	16-68
16.7. Update and maintenance	16-69
Update and maintenance	16-70
Update Strategy for WebSphere Application Server V8.5.5	16-71
Verify current version levels and applied updates	16-72
Fix pack download and update	16-73
Installing maintenance packages (1 of 2)	16-74
Installing maintenance packages (2 of 2)	16-75
Before contacting IBM support	16-76
Backing up Installation Manager	16-77
Unit summary	16-78
Checkpoint questions	16-79
Checkpoint answers	16-80

Unit 17. Intelligent Management problem determination and problem determination tools	17-1
Unit objectives	17-2
Topics	17-3
17.1. Intelligent Management overview	17-5
Intelligent Management overview	17-6
Intelligent Management	17-7
Performance Management	17-9
Health Management	17-10
Application Edition Management	17-11
Intelligent routing	17-13
Intelligent Management components	17-14
Dynamic clusters	17-15
Service policies	17-16
Autonomic managers and services	17-17
Intelligent routers: the ODR	17-19
Intelligent routers: WebSphere plug-in	17-21
17.2. Intelligent Management problem determination	17-23

Intelligent Management problem determination	17-24
Components to watch	17-25
Examining on-demand configuration	17-26
On-demand router: Debugging a routing failure	17-27
ODR: Debugging 404 errors	17-28
ODR: Debugging 503 errors	17-29
ODR: Understanding the unreachable error	17-30
Application placement controller issues (1 of 2)	17-31
Application placement controller issues (2 of 2)	17-32
17.3. Intelligent Management problem determination tools	17-33
Intelligent Management problem determination tools	17-34
Problem determination tools	17-35
Centralized logging: Simplifying log collection	17-36
Centralized logging: simplifying log collection	17-37
Request based tracing	17-38
Request based tracing	17-39
Health Management overview	17-40
Health management as part of monitoring solution	17-41
Health policies	17-42
Predefined health policy conditions	17-43
Predefined health policy conditions (1 of 2)	17-44
Predefined health policy conditions (2 of 2)	17-45
Custom health policy conditions	17-46
Custom health policy conditions	17-47
Predefined health policy actions	17-48
Predefined health policy actions	17-49
Custom health policy actions (1 of 2)	17-50
Custom health policy actions (2 of 2)	17-51
Health policy reaction mode	17-52
Cluster restart constraints	17-53
Unit summary	17-54
Checkpoint questions	17-55
Checkpoint answers	17-56
Exercise 13	17-57
Exercise objectives	17-58
Unit 18. Course summary	18-1
Unit objectives	18-2
Course learning objectives (1 of 2)	18-3
Course learning objectives (2 of 2)	18-4
Class evaluation	18-5
References	18-6
References	18-7
Unit summary	18-8
List of abbreviations	X1

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, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

AFS™	AIX 6™	AIX®
Approach®	DB™	DB2®
developerWorks®	HACMP™	IMS™
MVS™	PartnerWorld®	RACF®
Rational®	RDN®	Redbooks®
Redpaper™	Tivoli®	WebSphere®
WPM®	z/OS®	

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

•

Course description

WebSphere Application Server V8.5.5 Problem Determination

Duration: 5 days

Purpose

This 5-day instructor-led course teaches you how to manage WebSphere Application Server problems more skillfully within your organization by using problem determination tools and techniques. The instructor and students explore common scenarios that you might face in your daily activities. You also learn methodologies and techniques for problem determination, including how to use online IBM support tools to resolve problems. In addition, you learn how to communicate more effectively with IBM support teams so they can identify a problem and find its solution.

The course covers problems that are associated with Java virtual machine (JVM) tuning and memory management, database connectivity, connection pool configuration, security configuration, server start and stop failures, application deployment, web requests, and default messaging.

In hands-on lab exercises throughout the course, you gain practical experience with problem determination techniques by using your newly acquired skills within various scenarios. These scenarios include hung threads, OutOfMemory errors, crashes, data source configuration, security-related issues, server start and stop failures, web requests, and Java Message Service (JMS) message flow issues.

Audience

This course is designed for anyone who works on WebSphere related applications and projects, including administrators, IBM Business Partners, independent software vendors (ISVs), and consultants.

Prerequisites

Before taking this course, you should have basic operating skills for the Linux or UNIX operating systems. You should also have WebSphere administration skills, which can be obtained by completing an IBM WebSphere Application Server V8.5.5 Administration course (WA855, VA855, or ZA855) or through practical experience in administering a WebSphere Application Server environment.

Objectives

After completing this course, you should be able to:

- Use IBM Support Assistant to organize and analyze problem artifacts
- Use problem determination techniques to identify common problems
- Apply problem investigation approaches such as analysis and isolation
- Gather diagnostic data problem artifacts by using administrative tools
- Troubleshoot JVM-related problems such as hung threads, out of memory issues, and crashes
- Use IBM Support Assistant to run tools that analyze diagnostic data
- Identify and troubleshoot common problems with database connections
- Configure and tune database connection pools
- Troubleshoot WebSphere security problems that are associated with authentication, authorization, SSL, and Java 2 policies
- Identify and resolve Java EE application deployment problems
- Troubleshoot HTTP request flow problems from web server to web container
- Identify and resolve application server startup failures
- Troubleshoot problems that are associated with WebSphere default messaging and SI bus
- Troubleshoot WebSphere installation problems
- Use Intelligent Management features to configure health policies and tasks
- Communicate effectively with IBM support teams

Curriculum relationship

- WA855, VA855, ZA855
- WA715, VA715

Agenda

Day 1

- Course introduction
- Unit 1: Overview of WebSphere Application Server systems and components
- Unit 2: Using the IBM Support Assistant Team Server 5.0
- Exercise 1: Using the IBM Support Assistant Team Server 5.0
- Unit 3: Problem determination methods
- Unit 4: Gathering diagnostic data
- Exercise 2: Gathering diagnostic data

Day 2

- Unit 5: Introduction to JVM-related problems
- Exercise 3: Introduction to configuring garbage collection policies
- Unit 6: How to troubleshoot hangs
- Exercise 4: Troubleshooting hung threads
- Unit 7: How to troubleshoot crashes
- Exercise 5: Troubleshooting crashes

Day 3

- Unit 8: Introduction to WebSphere out-of-memory problems
- Exercise 6: Troubleshooting an out-of-memory condition
- Unit 9: Introduction to database connection problems
- Exercise 7: Troubleshooting database connection problems
- Unit 10: Tuning and connection pool management problems
- Exercise 8: Troubleshooting a connection leak

Day 4

- Unit 11: WebSphere security configuration problems
- Exercise 9: Troubleshooting security problems
- Unit 12: Application deployment problems
- Unit 13: Server start failures
- Exercise 10: Troubleshooting server start failures
- Unit 14: Request flow and web container problems
- Exercise 11: Troubleshooting request flow and web container problems

Day 5

- Unit 15: Default messaging provider problem determination
- Exercise 12: Troubleshooting WebSphere default messaging
- Unit 16: WebSphere installation problems when using IBM Installation Manager
- Unit 17: Intelligent Management problem determination and problem determination tools
- Exercise 13: Configuring health management policies
- Unit 18: Course summary

Unit 1. Overview of WebSphere Application Server systems and components

What this unit is about

This unit provides an overview of the topology of a system. It identifies and describes key components and troubleshooting points.

What you should be able to do

After completing this unit, you should be able to:

- Describe stand-alone server architecture
- Describe Network Deployment (ND) cell architecture
- List and describe the function of IBM products that are involved in implementing stand-alone and distributed architectures
- Identify the components of the application server and describe the services that they provide
- Identify the components of an ND cell and describe the function of each
- Describe the flexible management model
- Describe the various types of clients for the application server
- Describe the flow of an application request
- Describe the flow of security and administration requests
- Identify common troubleshooting points in the end-to-end flow of client requests

How you will check your progress

- Checkpoint questions

References

SG24-7971-00 WebSphere Application Server V8: *Administration and Configuration Guide*

Course WA380: *IBM WebSphere Application Server V8 Administration on Windows*

Course WA580: *IBM WebSphere Application Server V8 Administration on Linux*

Unit objectives

After completing this unit, you should be able to:

- Describe stand-alone server architecture
- Describe Network Deployment (ND) cell architecture
- List and describe the function of IBM products that are involved in implementing stand-alone and distributed architectures
- Identify the components of the application server and describe the services that they provide
- Identify the components of an ND cell and describe the function of each
- Describe the flexible management model
- Describe the various types of clients for the application server
- Describe the flow of an application request
- Describe the flow of security and administration requests
- Identify common troubleshooting points in the end-to-end flow of client requests

© Copyright IBM Corporation 2013

Figure 1-1. Unit objectives

WA5913.0

Notes:

Topics

- Components and services
- Network Deployment components
- Client request flow

© Copyright IBM Corporation 2013

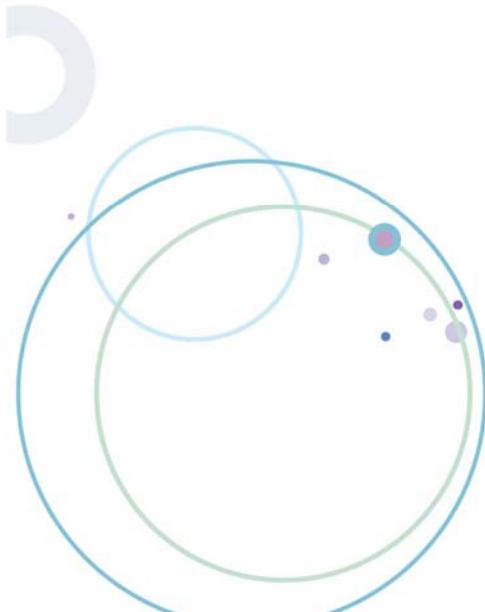
Figure 1-2. Topics

WA5913.0

Notes:

1.1. Components and services

Components and services



© Copyright IBM Corporation 2013

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

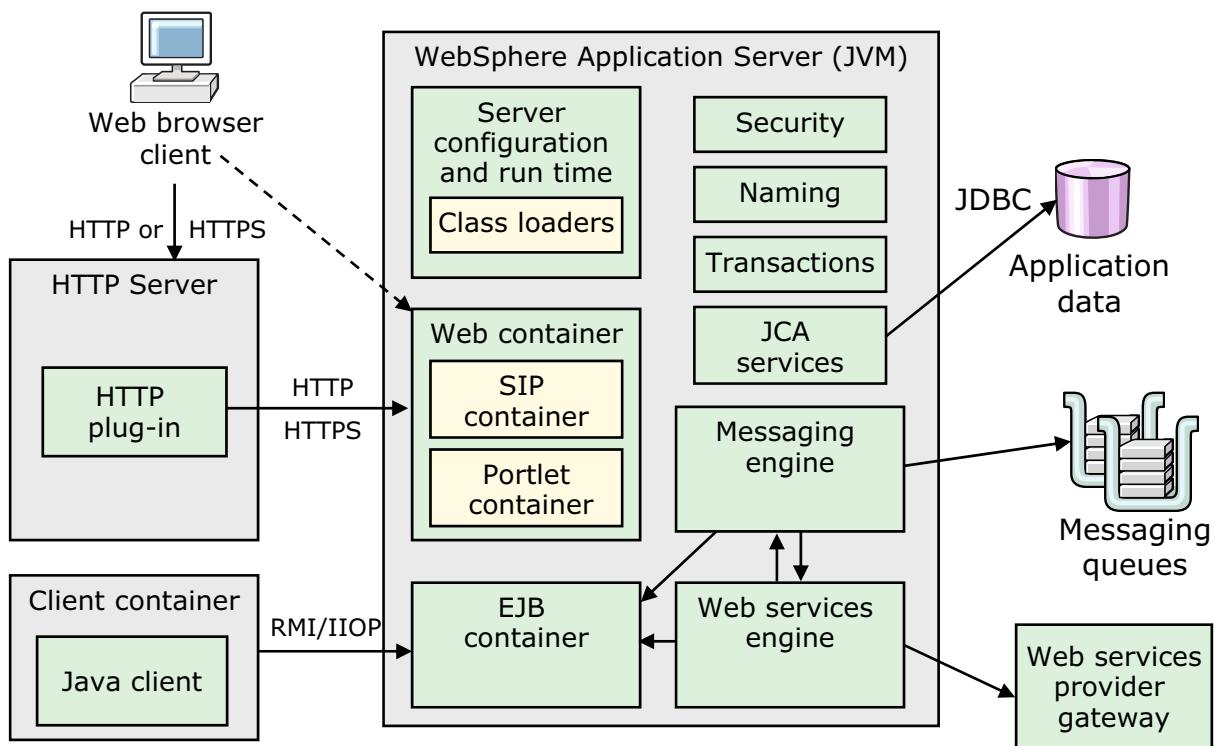
8.0

Figure 1-3. Components and services

WA5913.0

Notes:

WebSphere Application Server architecture



© Copyright IBM Corporation 2013

Figure 1-4. WebSphere Application Server architecture

WA5913.0

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 runs. If you are familiar with Java EE, you recognize the web container and the EJB container. The web container runs servlets and JavaServer Pages (JSPs), both of which are Java classes that generate web markup language. 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 JavaBeans (EJBs) to run 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. These EJBs return data to the web container or update the data on behalf of the servlet or JSP.

The JMS messaging engine is built into the application server. This engine is a pure Java messaging engine. JMS destinations, which are known as queues and topics, provide asynchronous messaging services to the code that is running inside the containers. JMS is covered in more depth later in this course.

As you see in more detail later on, the web services engine enables application components to be exposed as web services, which can be accessed with SOAP.

Several other services are run within the application server, including the dynamic cache, data replication, security, and others. These services are 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 for WebSphere to handle, and routes the requests to the appropriate server. The plug-in is critical to workload management of HTTP requests, as it can distribute the load to multiple application servers, and steer traffic away from unavailable servers. It too reads its configuration from a special XML file.

1.

-

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:
 - Web container transport chains
 - Servlet processing
 - JSP processing
 - HTML and other static content processing
 - Session management
 - Thread pool
 - Portlet container

© Copyright IBM Corporation 2013

Figure 1-5. Web container

WA5913.0

Notes:

- **Web container transport chains**

The web container inbound transport chain is used to direct requests to the web container. The chain consists of a TCP inbound channel that provides the connection to the network, and an HTTP inbound channel that serves HTTP 1.0 and 1.1 requests. The chain continues with 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; it 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**

The web container inbound chain serves requests for HTML and other static content that are directed to the web container. However, in most cases, the use of 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 programming interface (API) specification.

- **Web services engine**

Web services are provided as a set of APIs in cooperation with the Java EE applications. Web services engines are provided to support SOAP.

The following properties allow you to configure the services that the web container provides:

- **Default virtual host**

This virtual host is the default 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 so that you can use servlet caching.

- **Session management**

You can determine how the web container manages HTTP session data. This determination 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 that are used for accessing applications in the web container. By default, you have four transport chains predefined. These transport chains 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 in the communications section. Port numbers must be unique for each application server instance on a machine.

- **Custom properties**

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

Enterprise JavaBeans (EJB) container

- The Enterprise JavaBeans (EJB) container provides the runtime services that are needed to deploy and manage enterprise beans
- A server process that handles requests for:
 - Session beans
 - Entity beans
 - Message-driven beans (MDBs)
- The enterprise beans that are packaged in EJB modules and installed in an application server, do not communicate directly with the server
 - The EJB container provides an interface between the enterprise beans and the server
- Together, the EJB container and the application server provide the enterprise bean runtime environment
- The WebSphere run time provides many low-level services, including:
 - Thread pool support
 - Transaction support
- The EJB container manages data storage and retrieval for the contained EJBs
- A single EJB container can host more than one EJB application module

© Copyright IBM Corporation 2013

Figure 1-6. Enterprise JavaBeans (EJB) container

WA5913.0

Notes:

The following properties allow you to configure the services that the EJB container provides:

- **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 must 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 whether 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-compliant CMP beans.

- **Initial state**

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

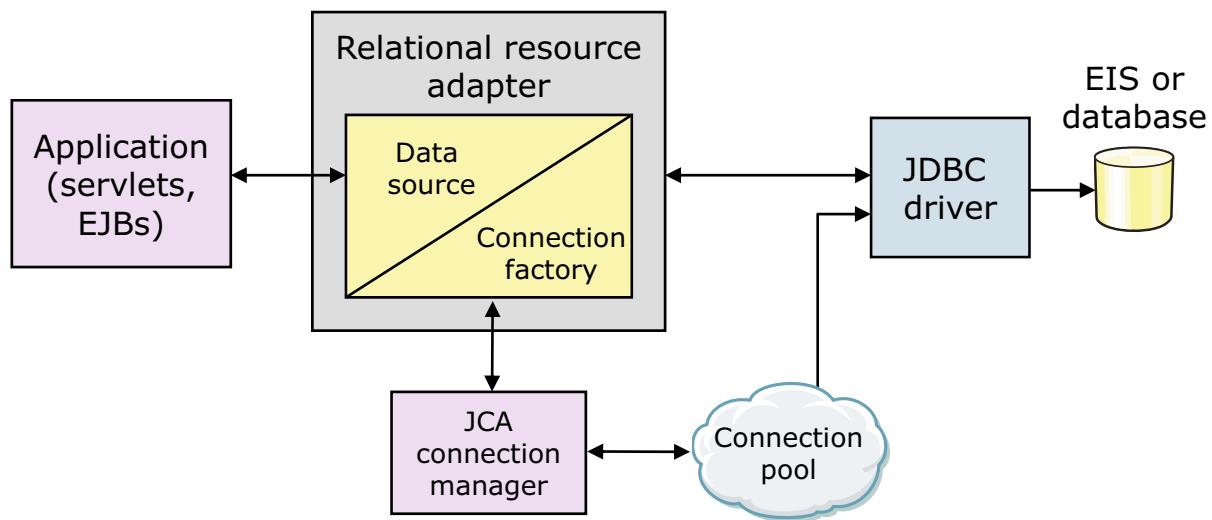
- **EJB cache settings**

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. It reduces 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. WebSphere uses this attribute to determine how large the cache is and when to remove components from the cache to reduce its size.

Java EE Connector Architecture service (JCA)

- The JCA Connection Manager administers:
 - Connections that are obtained through resource adapters that the JCA defines
 - Data sources that the JDBC 2.0 Extensions define



© Copyright IBM Corporation 2013

Figure 1-7. Java EE Connector Architecture service (JCA)

WA5913.0

Notes:

Connection management for access to enterprise information systems (EIS) in WebSphere Application Server is based on the Java EE Connector Architecture (JCA) specification, also sometimes referred to as J2C. The connection between the enterprise application and the EIS is made by using 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 that are defined according to the JCA specification and sources that are defined according to 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 that use 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 that EIS vendors or other third-party vendors supply. It provides the connectivity between Java EE 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.

Transaction service

- WebSphere applications use transactions to coordinate multiple updates to resources as one unit of work
- Applications, or the container, start and end transactions; 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
- The transaction service writes information to the **transaction log** for every global transaction that involves two or more resources, or that is distributed across multiple servers
- You can also configure WebSphere applications to interact with:
 - Databases
 - Java Message Service (JMS) queues
 - JCA connectors
 - EJBs in other application servers

© Copyright IBM Corporation 2013

Figure 1-8. Transaction service

WA5913.0

Notes:

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

- A session bean can use either 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.

Data replication service

- The data replication service (DRS) is responsible for replicating in-memory data among WebSphere processes
 - Allows for high availability and failover recovery
 - Improves performance and scalability
- Data replication service is used 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

© Copyright IBM Corporation 2013

Figure 1-9. Data replication service

WA5913.0

Notes:

Replication domains, consisting of server or cluster members that must 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 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 servers 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.

Name service

- Each application server hosts a name service that provides a Java Naming and Directory Interface (JNDI) namespace
- Registers all EJB and Java EE resources that the application server hosts, 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

© Copyright IBM Corporation 2013

Figure 1-10. Name service

WA5913.0

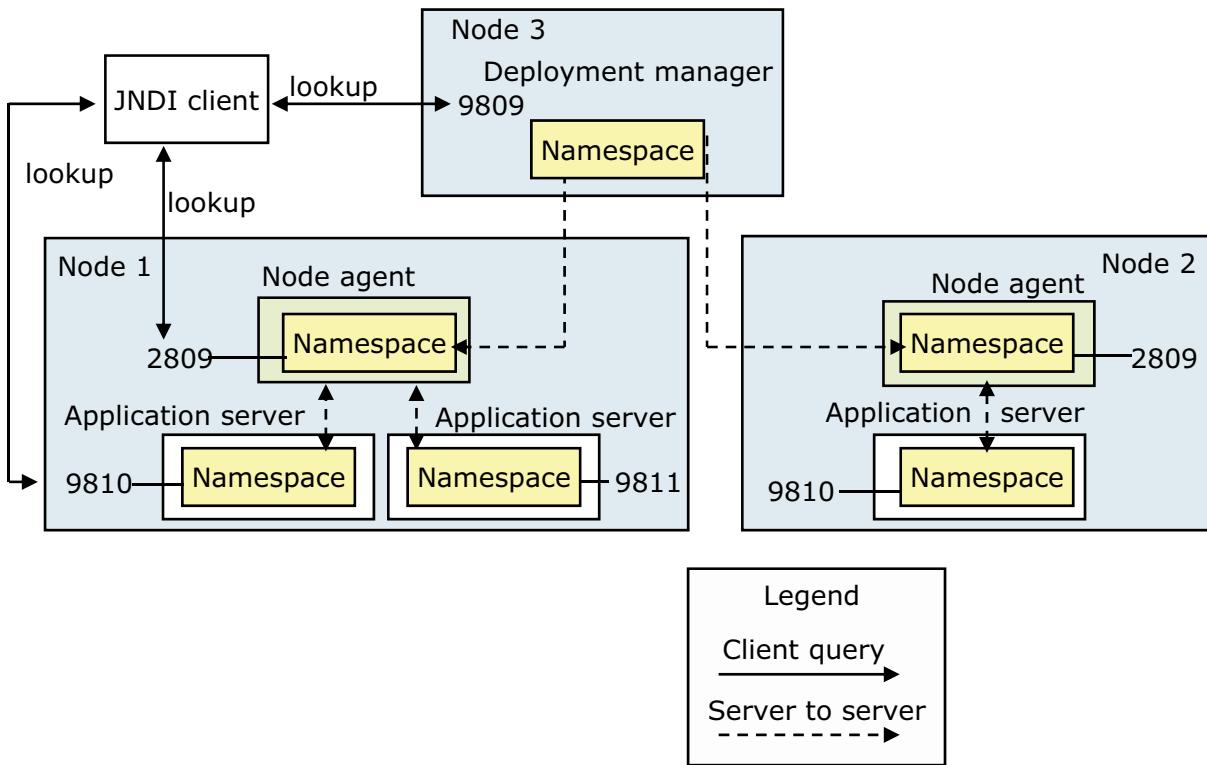
Notes:

The naming service is used to register resources that the application server hosts. 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 that is stored in CosNaming and interacts with the CosNaming server on behalf of the client.

Clients of WebSphere applications use the naming architecture 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 that is bound with that name. The namespace can be accessed and manipulated through a name server.

Naming topology



© Copyright IBM Corporation 2013

Figure 1-11. Naming topology

WA5913.0

Notes:

Distributed namespace:

For more 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. A cell-persistent root is used for cell-scoped persistent bindings, and a node-persistent root is used to bind objects with a node scope.

Federated namespace structure:

A *namespace* is a collection of all names that are bound to a particular name server. A namespace can contain naming context bindings to contexts in other servers. If it does, then the namespace is said to be a *federated namespace*, because it is a collection of namespaces 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 not apparent 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, and the various server roots and persistent partitions of the namespace are interconnected through 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 according to 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.

dumpNameSpace tool

You can use the *dumpNameSpace* tool to view the namespace for a particular server. The *dumpNameSpace* tool does not dump all of the objects in the distributed namespace. It dumps only the objects that are in the local namespace of the process against which the command was run.

Object Request Broker service

- Using Internet Inter-ORB Protocol (IIOP), an Object Request Broker (ORB), manages the interaction between EJB clients and EJBs
 - ORB service enables clients to make requests and receive responses from EJBs in a network-distributed environment
 - Provides a framework for clients to locate objects in the network and call operations on those objects
- The client-side ORB:
 - Creates an IIOP request that contains the operation and any required parameters for sending the request across the network
- The server-side ORB:
 - Receives the IIOP request, locates the target object, starts the requested operation, and returns the results to the client
- **Note:** Web services and JMS messaging start EJBs when EJBs are wrapped in SCA components

© Copyright IBM Corporation 2013

Figure 1-12. Object Request Broker service

WA5913.0

Notes:

An Object Request Broker (ORB) uses Internet Inter-ORB Protocol (IIOP) to manage the interaction between clients and servers. 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 in the same running process as the client. The ORB service provides location transparency. The client calls an operation on a local object, which is known as a *stub*. Then, the stub forwards the request to the intended 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 unmarshals the returned results and passes the result to the stub, which returns the result to the client application, as though the operation were run locally.

When using the SCA programming model, IIOP communication is no longer the only way to invoke EJBs. When they are wrapped in SCA components, they can be invoked in various ways, including web services and JMS messaging.

WebSphere default messaging

- Integrated asynchronous capabilities for WebSphere
 - Integral JMS messaging service for WebSphere Application Server
 - Fully compliant JMS 1.1 provider
 - JMS provider is the default messaging provider
- Based on service integration bus (SIBus) technology
 - Intelligent infrastructure for service-oriented integration
 - Unifies SOA, messaging, message brokering, and publish/subscribe
- Complements and extends WebSphere MQ and application server
- Other WebSphere family products use default messaging

© Copyright IBM Corporation 2013

Figure 1-13. WebSphere default messaging

WA5913.0

Notes:

The service integration functionality within WebSphere Application Server provides a highly flexible messaging fabric that supports a service-oriented architecture with a wide spectrum of quality of service options, supported protocols, and messaging patterns. It supports both message-oriented and service-oriented applications.

SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work that a service provider does to achieve specified results for a service consumer. Both provider and consumer are roles that software agents play on behalf of their owners.

Security service

- Each application server hosts a security service
- The security service uses the security settings that are stored in the configuration repository to provide:
 - Authentication and authorization
 - SSL encryption
 - Java 2 security
- User registries that contain authentication data can be configured with one of the following repositories:
 - File-based (default)
 - Local OS
 - LDAP server
 - Custom user registry
 - Federated repository - combination of file-based, LDAP servers, CUR, or databases

© Copyright IBM Corporation 2013

Figure 1-14. Security service

WA5913.0

Notes:

Each application server JVM hosts a security service. The security service uses the security settings that are held in the configuration repository to provide authentication and authorization functionality.

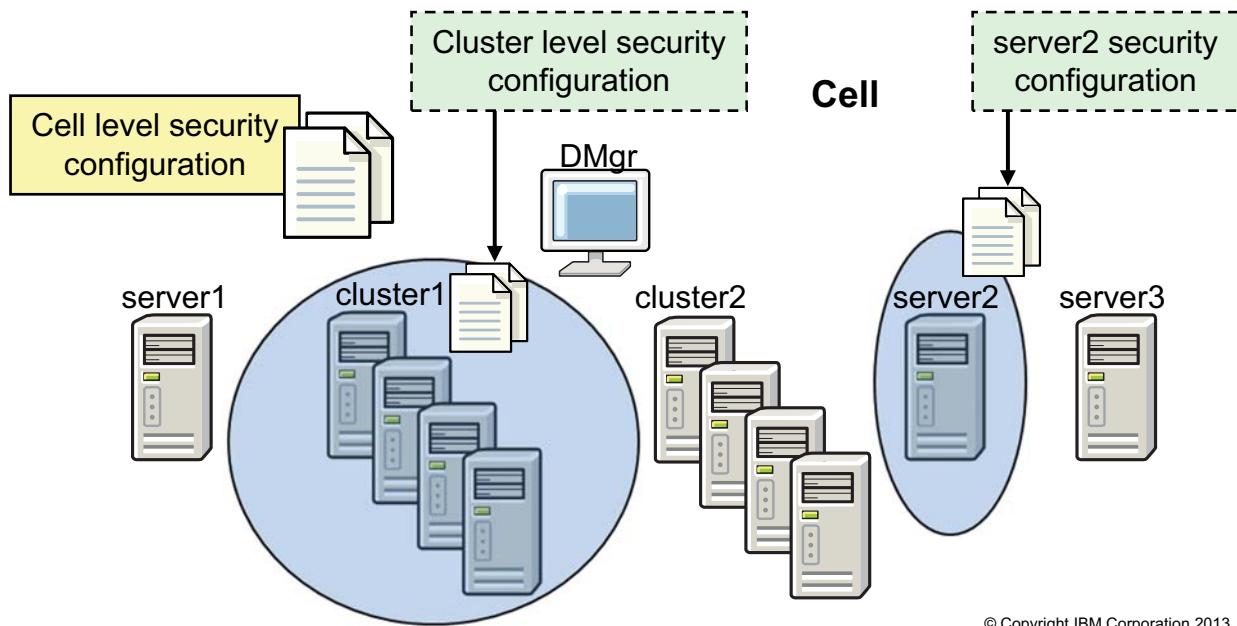
User registries that contain authentication data can be configured by using the local OS, an LDAP directory server, a custom user registry, or a federated repository.

Federated repositories can comprise multiple user registries in LDAP servers, flat files, and databases.

The security service can also be used to enable SSL encryption between WebSphere clients and servers.

Security domains

- With security domains, it is possible to not only have a cell level security configuration, but also have multiple other security configurations that are scoped at different levels



© Copyright IBM Corporation 2013

Figure 1-15. Security domains

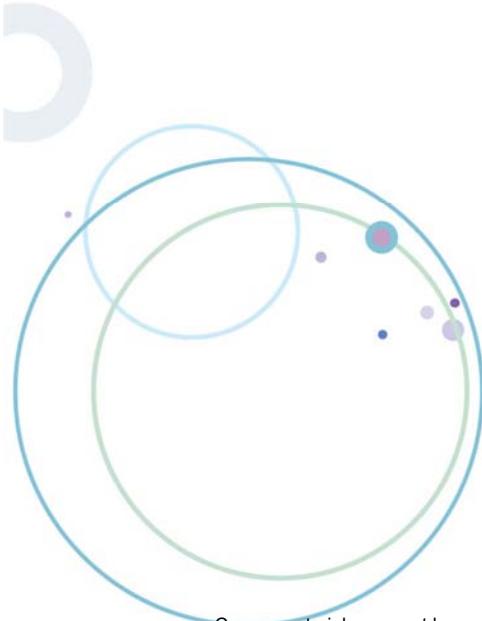
WA5913.0

Notes:

Security domains allow security to be defined at multiple levels, not just at the cell level. With security domains, it is possible to define one set of security settings for one application server and another set of configurations for a second application server. Only users who are assigned to the administrator role can configure multiple security domains. For example, with security domains, it is possible to have different user registries that are configured for distinct parts of the cell.

1.2. Network Deployment components

Network Deployment components



© Copyright IBM Corporation 2013

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

8.0

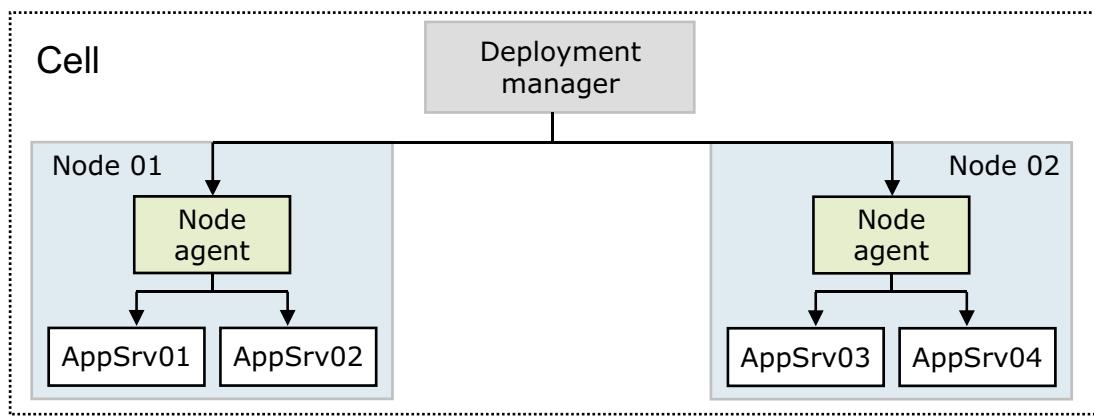
Figure 1-16. Network Deployment components

WA5913.0

Notes:

Network Deployment concepts

- A **deployment manager** process manages the node agents
 - Holds the configuration repository for the entire management domain, called a **cell**
 - In a cell, the administrative console runs inside the deployment manager
- A **node** is a logical grouping of application servers
 - A single **node agent** process manages each node
 - Through profile configuration, multiple nodes can exist on a single computer



© Copyright IBM Corporation 2013

Figure 1-17. Network Deployment concepts

WA5913.0

Notes:

The deployment manager here is an application server that manages the administrative environment within a cell. A profile represents a node. Multiple nodes can exist on a single machine by using profiles. The node agent is an important process that allows for communication of administrative information (commands and configuration files) to reach the application servers.

Runtime flow

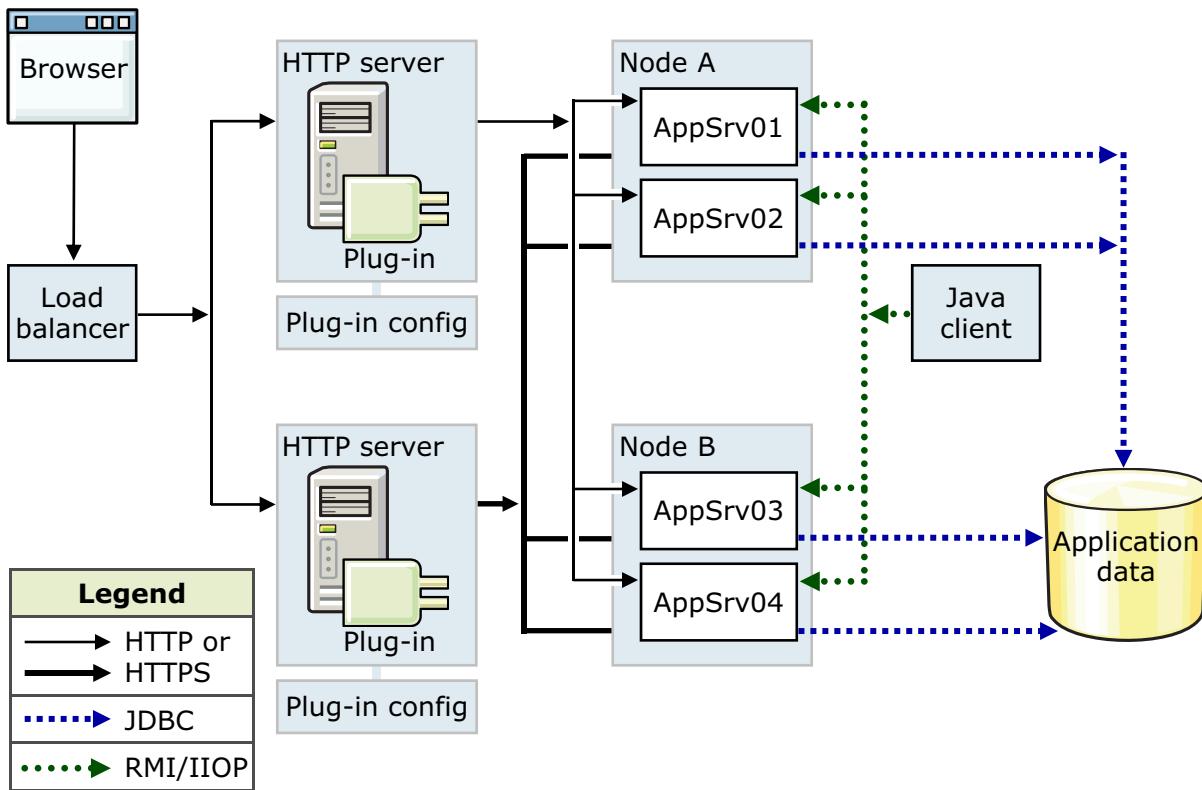


Figure 1-18. Runtime flow

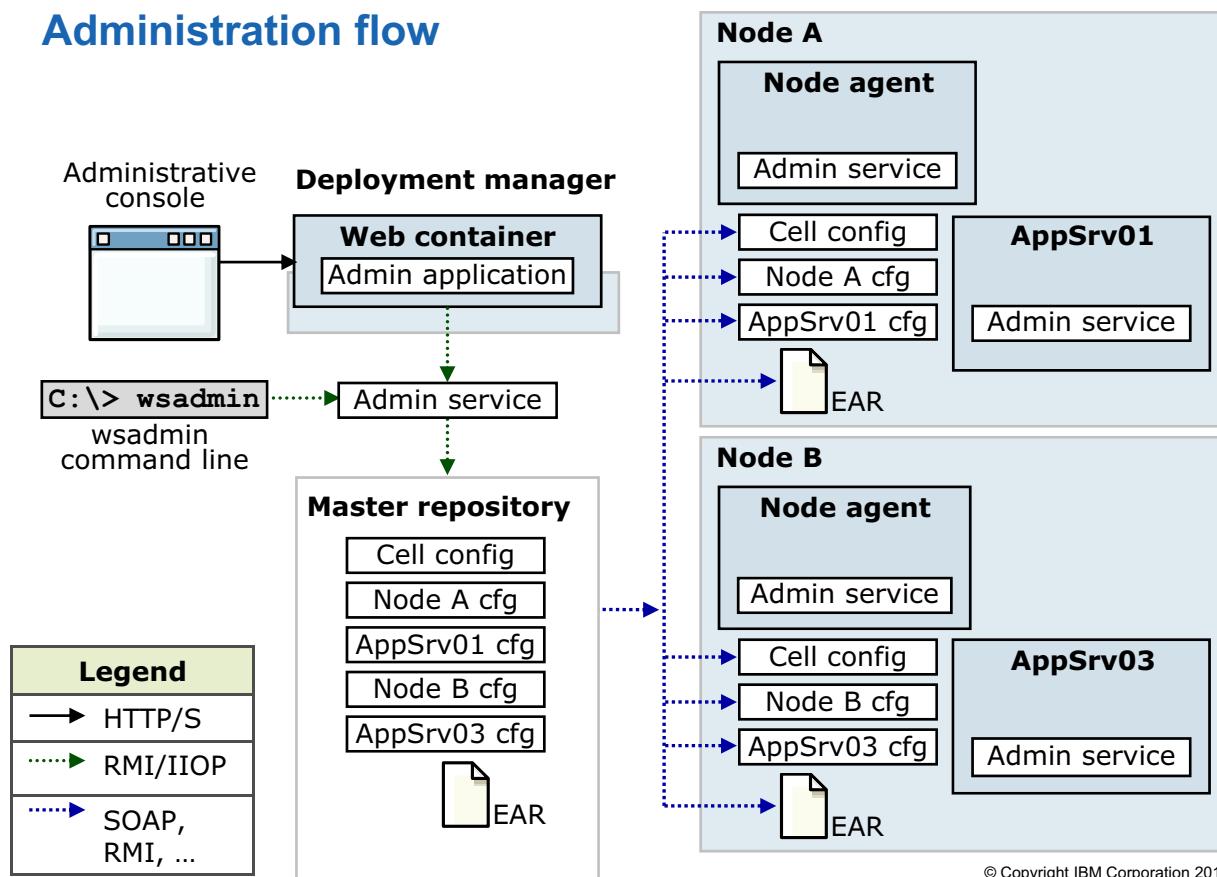
WA5913.0

Notes:

The main theme with Network Deployment is distributed applications. While the “flow” of an application remains the same, there are significant additions to the runtime of an application. Note the “load balancer”: it allows for multiple HTTP servers. Users point their browsers to the load balancer, and their requests are workload managed to an HTTP server. As soon as a request reaches one of these HTTP servers, the HTTP server plug-in load balances the request between the application servers that it is configured to serve. When the request enters the application server, the flow is identical to how it was in Express and Base.

The Java client’s requests to EJBs can also be workload managed so that the requests do not all go to one application server.

Administration flow



© Copyright IBM Corporation 2013

Figure 1-19. Administration flow

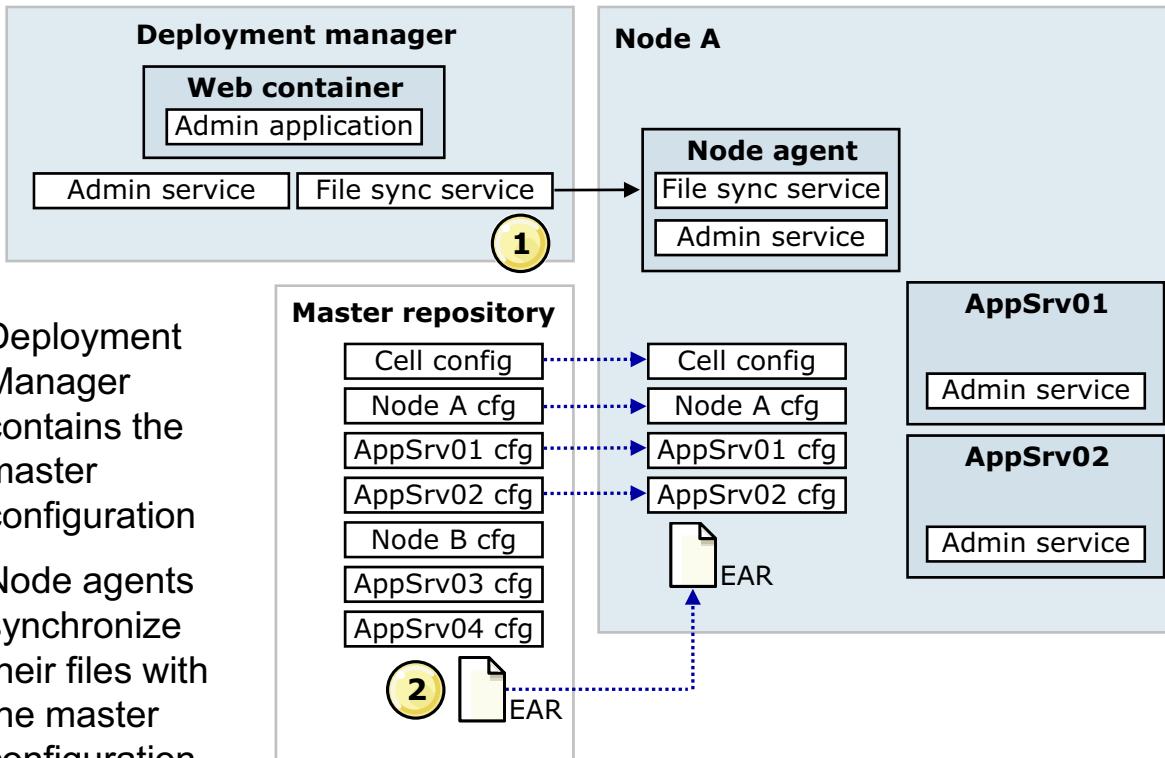
WA5913.0

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 communication flow 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, which are associated with the deployment manager. All updates to the configuration files should go through the deployment manager. You are going to see in a moment how this process works. Be 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. The configuration files overwrite them from the master files.

File synchronization and file transfer



© Copyright IBM Corporation 2013

Figure 1-20. File synchronization and file transfer

WA5913.0

Notes:

File synchronization service

Node agents synchronize their files with the “master” configuration (repository) as follows:

- Automatically at startup
- Periodically through configuration
- Manually from the administrative console or command line

During synchronization:

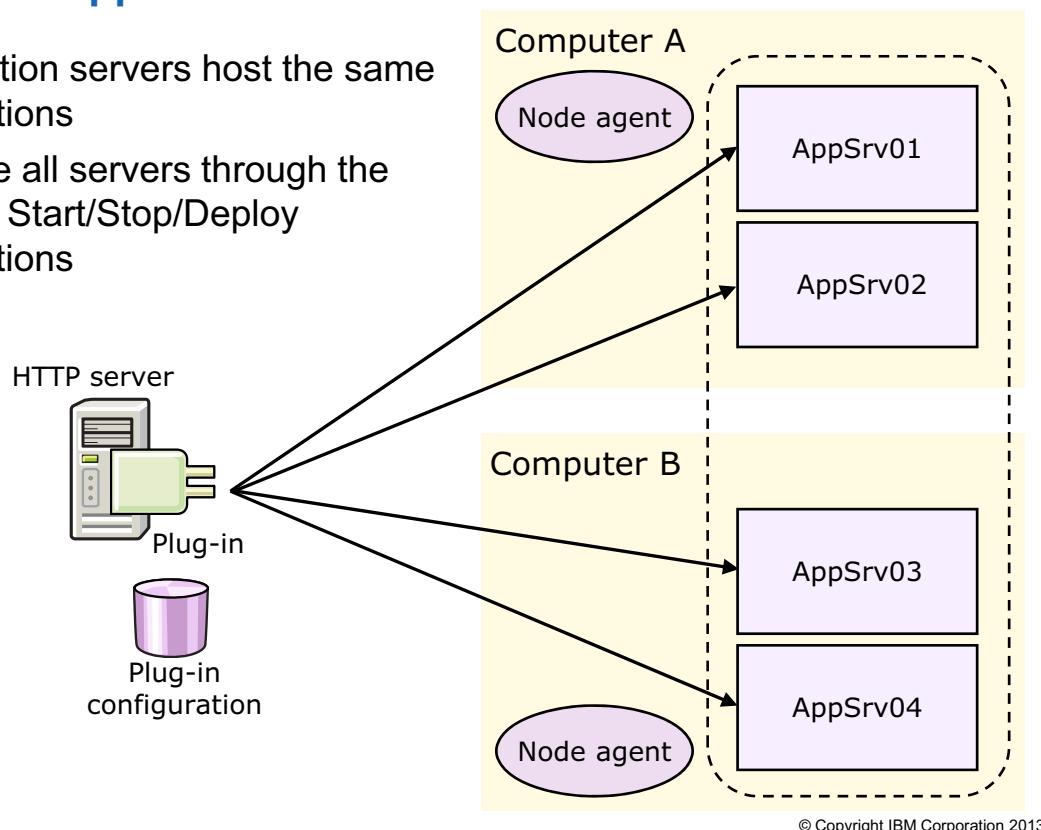
- The node agent checks for changes to master configuration
- New or updated files are copied to the node

This administrative service is 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 are 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 whether any files that apply to the node are 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.

Clustered application servers

- Application servers host the same applications
- Manage all servers through the cluster: Start/Stop/Deploy applications



© Copyright IBM Corporation 2013

Figure 1-21. Clustered application servers

WA5913.0

Notes:

A cluster of application servers provides scalability, throughput, and failover. The plug-in routes web requests for the HTTP server to servers in the cluster by using a weighted or random round-robin algorithm. The plug-in configuration file contains information about clusters, members of the cluster (clones), and the applications that the servers in the cluster host. In addition, each cluster member has a weight that is stored in the configuration file. The plug-in uses this weight in its routing algorithms to determine what percentage of incoming requests are routed to each cluster member.



Web servers

- Web servers can be defined to the administrative service as web server nodes, allowing Java EE applications to be associated with one or more defined web servers
- Each web server must have a web server plug-in that is installed, which forwards requests to the application servers
- Web server nodes can be managed or unmanaged

© Copyright IBM Corporation 2013

Figure 1-22. Web servers

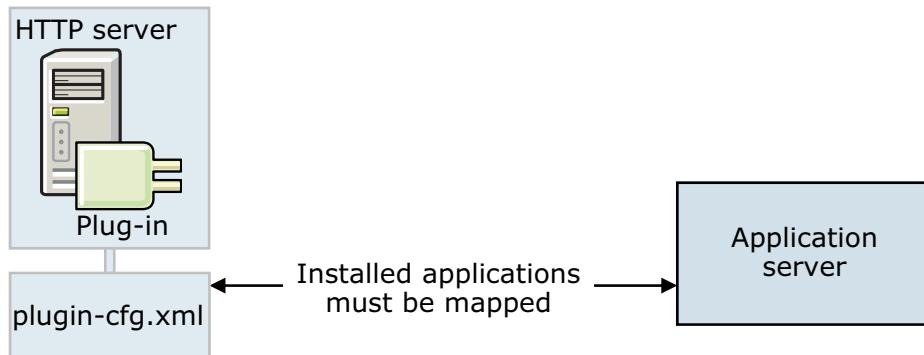
WA5913.0

Notes:

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

Web server plug-ins

- When a web request requires dynamic content, such as JSP or servlet processing, it must be forwarded to the application server
- The plug-in uses the configuration file to determine whether a request is routed to the web server or an application server
- Forwards the request to the appropriate web container
 - Forwards a request that is based on its URI
 - The plug-in can use HTTP or HTTPS to transmit the request
- Can load balance among cluster members using `plugin-cfg.xml` file



© Copyright IBM Corporation 2013

Figure 1-23. Web server plug-ins

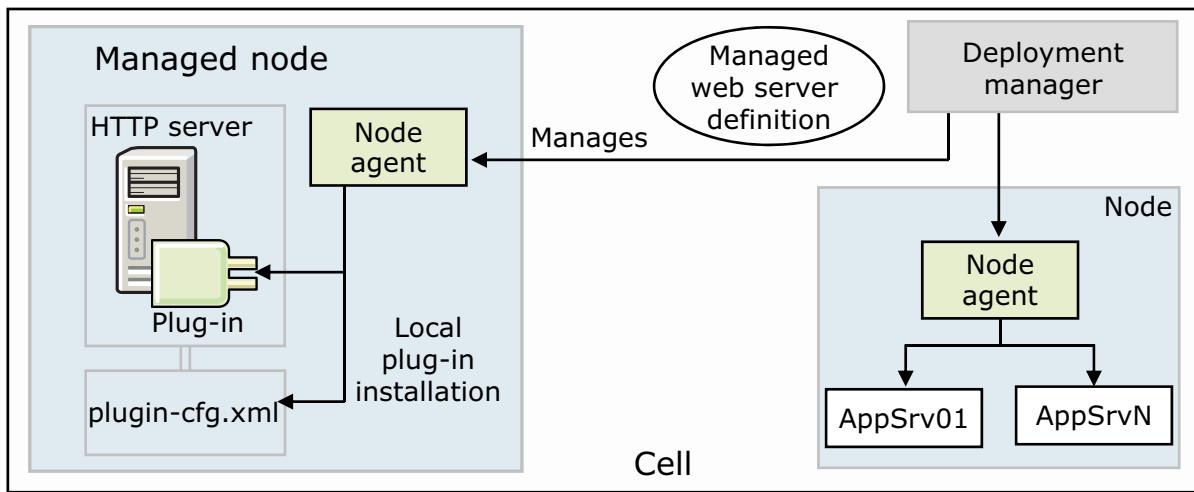
WA5913.0

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, which is on the WebSphere Application Server, to the web server plug-in directory. The plug-in uses the configuration file to determine whether the web server or an application server should handle the request. When a web 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.

Web servers: Managed nodes

- Managed nodes have a node agent on the web server computer that allows the deployment manager to administer the web server
- Administrator 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 plug-in configuration file to the web server



© Copyright IBM Corporation 2013

Figure 1-24. Web servers: Managed nodes

WA5913.0

Notes:

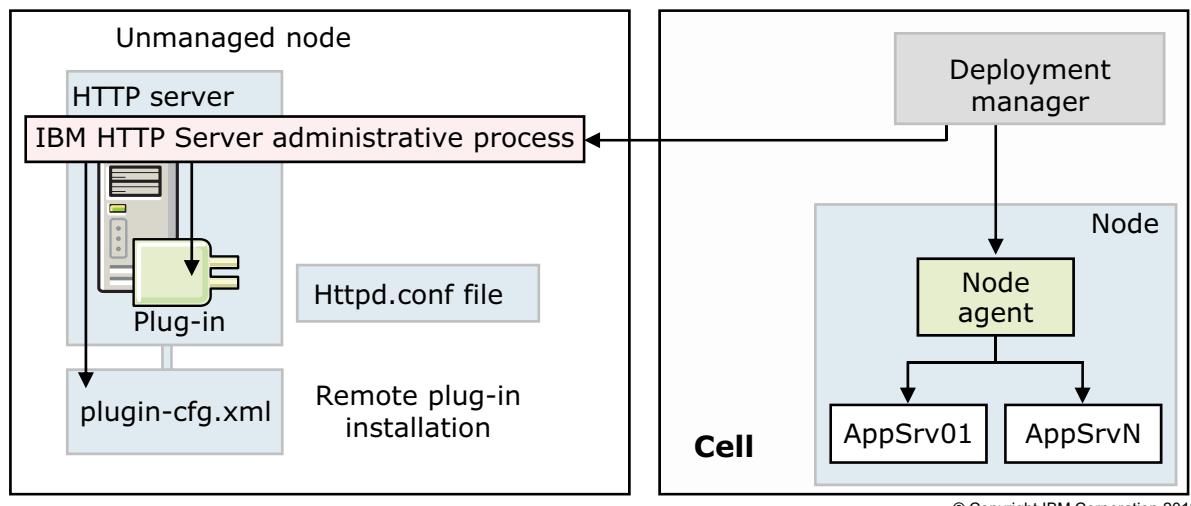
If the web server is an IBM HTTP Server and the IBM HTTP Server administration server is installed and properly configured, you can also:

- Display the IBM HTTP Server error log (`error.log`) and access log (`access.log`) files
- Display and edit the IBM HTTP Server configuration file (`httpd.conf`)

Managed web server nodes are usually behind the firewall with WebSphere Application Server installations.

Web servers: Unmanaged nodes

- WebSphere Application Server does not manage unmanaged web server nodes
 - Normally found outside the firewall, or in the DMZ
 - You must either manually copy, or FTP, web server plug-in configuration files to the web server
 - If you define the web server as a node, you can generate custom plug-in configuration files for it



© Copyright IBM Corporation 2013

Figure 1-25. Web servers: Unmanaged nodes

WA5913.0

Notes:

You cannot propagate an updated plug-in configuration file to a non-IBM HTTP Server that is defined to an unmanaged node. You must manually install an updated plug-in configuration file to a web server that is defined to an unmanaged node. Web servers that are defined to an unmanaged node are typically remote web servers. Remote web servers are web servers that are not on the same machine as the application server.

High availability (HA) manager

- A high availability manager provides a framework that allows singleton services to make themselves highly available
- Examples of singleton services that use this framework include:
 - Transaction managers for cluster members
 - Default messaging provider message engines
- HA manager provides a mechanism that allows servers to easily exchange state data
 - The workload management (WLM) component uses this mechanism to build and maintain a highly available routing table
- HA manager provides a special framework for high speed and reliable messaging between processes
 - The data replication service uses this framework when the product is configured for memory-to-memory replication
- An HA manager instance runs on every server in a cell
 - Application server, proxy server, node agent, and deployment manager

© Copyright IBM Corporation 2013

Figure 1-26. High availability (HA) manager

WA5913.0

Notes:

A high availability manager instance runs on every application server, proxy server, node agent, and deployment manager in a cell. A cell can be divided into multiple high availability domains that are known as core groups. Each high availability manager instance establishes network connectivity with all other high availability manager instances in the same core group by using a specialized, dedicated, and configurable transport channel. The transport channel provides mechanisms that allow the high availability manager instance to detect when other members of the core group start, stop, or fail.

Within a core group, high availability manager instances are elected to coordinate high availability activities. An instance that is elected is known as a core group coordinator. The coordinator is highly available, so that if a process that serves as a coordinator stops or fails, another instance is elected to assume the coordinator role, without loss of continuity.

Highly available components

A *highly available component* is a component for which a high availability group is defined on the processes where that component can run. The coordinator tracks high availability group membership, and knows on which processes each highly available component can run.

The coordinator also associates a high availability policy with each high availability group. A *high availability policy* is a set of directives that aid the coordinator in managing highly available components. For example, a directive might specify that a component runs on a specific process, if that process is available. Directives are configurable, which makes it possible for you to tailor policies to your installation.

The coordinator is notified as core group processes start, stop, or fail and knows which processes are available at any specific time. The coordinator uses this information, along with the high availability group and policy information, to ensure that the component keeps functioning. The coordinator uses the policy directives to determine on which process it starts and runs each component. If the chosen process fails, the coordinator restarts the component on another eligible process. This process reduces the recovery time, automates failover, and eliminates the need to start a replacement process.

State data exchange

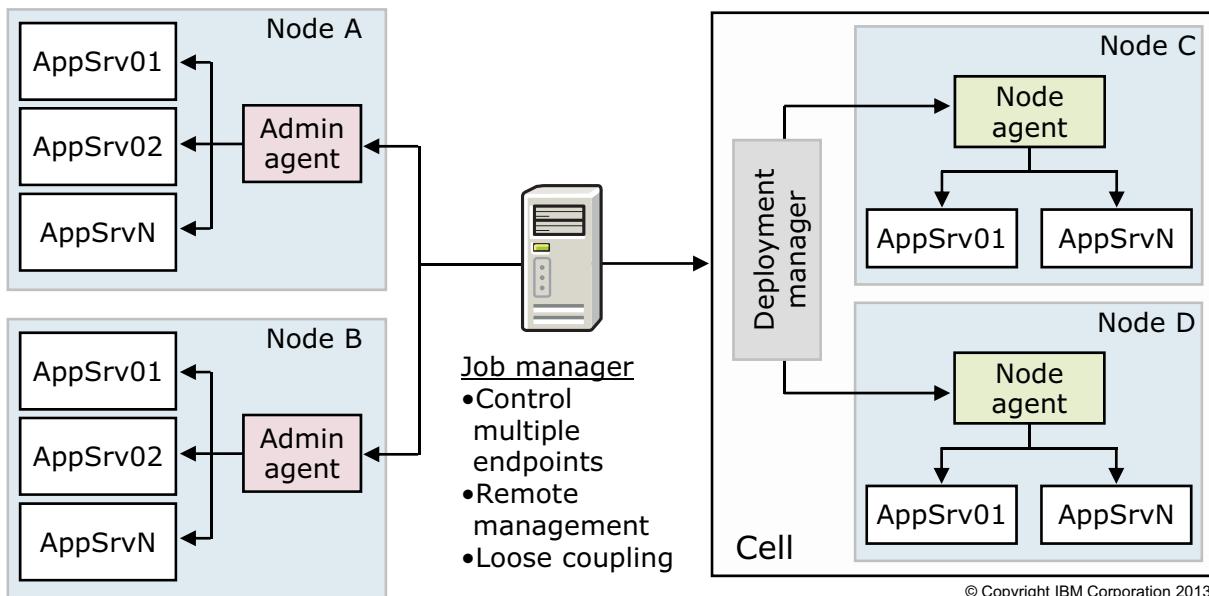
The high availability manager provides a specialized messaging mechanism that enables processes to exchange information about their current state. Each process sends or posts information that is related to its current state, and can register to be notified when the state of another process changes. The workload management (WLM) component uses this mechanism to build and maintain routing table information. Routing tables that are built and maintained by using this mechanism are highly available.

Replication

The data replication service (DRS) that is provided with the product is used to replicate HTTP session data, stateful EJB sessions, and dynamic cache information among cluster members. When DRS is configured for memory-to-memory replication, the transport channels that are defined for the high availability managers are used to pass this data among the cluster members.

Flexible management

- Loose management coupling
- Coordinates management across a group of endpoints
 - One job to install application across a number of nodes
- Can manage through administrative agent or deployment manager



© Copyright IBM Corporation 2013

Figure 1-27. Flexible management

WA5913.0

Notes:

Some additional management options not previously available to administrators include:

- Management of multiple base servers, up to a server farm that contains hundreds of base servers
- Coordinate management actions across multiple deployment managers

Base Application Server:

- Programming model
- QoS
- Security
- Administration

Network Deployment cell:

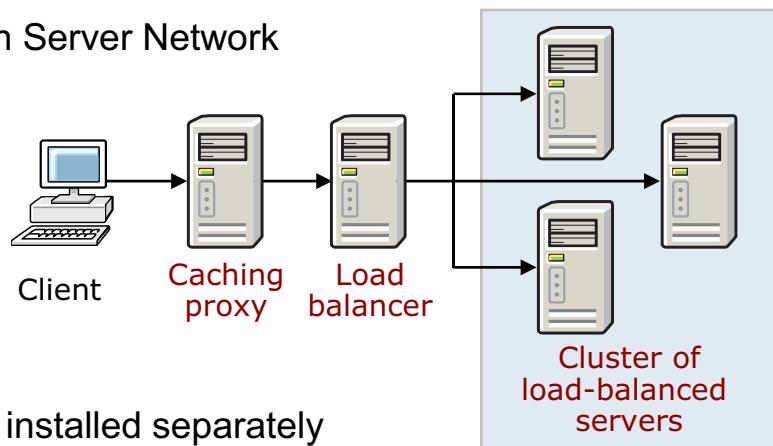
- Administration
- Clustering

Flexible management environments rely on asynchronous processing of work units (known as jobs) from the job manager. This processing lends itself to large scaling and can support many application servers without degrading performance. It also reduces latency and bandwidth requirements on the network; even dial-up lines to remote sites can work well without slowing down the overall system. Additionally, configuration information does not exist beyond the node level so that no bottleneck is associated with accessing a master configuration repository. Flexible management is not a replacement for the network deployment model. However, it can be used as an alternative to network deployment, and a job manager can combine the two by coordinating management actions across multiple deployment managers.

Flexible management is not available for z/OS.

Edge Components

- WebSphere Application Server Network Deployment package contains the following Edge Components functions:
 - Load balancer
 - Caching proxy
- Edge Components are installed separately from WebSphere Application Server
- **Load balancer** is responsible for balancing the load across multiple servers that can be within either local area networks or wide area networks
- Purpose of **caching proxy** is to reduce network congestion within an enterprise by offloading security and content delivery from web servers and application servers



© Copyright IBM Corporation 2013

Figure 1-28. Edge Components

WA5913.0

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 cacheable 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.

Liberty profile

- The Liberty profile provides a lightweight server with a small memory footprint
- Using WebSphere Application Server Developer Tools for Eclipse, you can create a server configuration quickly
- The simple and flexible configuration is stored in the `server.xml` file and contains the configuration for the runtime instance
- You can edit the `server.xml` file directly by using an XML editor or an Eclipse-based editor
 - The configuration lists the features (capabilities or bundles) that are installed in the server
 - By defining just the features that you need, the Liberty profile provides the smallest runtime footprint for applications
- This dynamic run time allows the adding of features and updating of configuration parameters without requiring you to restart the server

© Copyright IBM Corporation 2013

Figure 1-29. Liberty profile

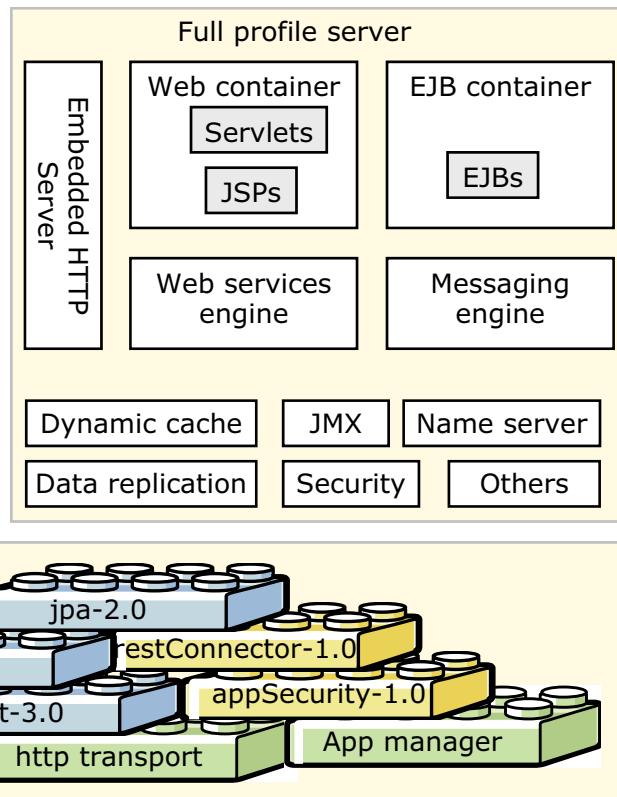
WA5913.0

Notes:

The Liberty profile provides a lightweight server with a small memory footprint. Using WebSphere Application Server Developer Tools for Eclipse, you can create a server configuration quickly with just a couple of mouse clicks. The simple and flexible configuration is stored in the `server.xml` file and contains the configuration for the runtime instance. You can edit the `server.xml` file directly by using an XML editor or an Eclipse-based editor. The configuration lists the features (capabilities or bundles) that are installed in the server. By defining just the features that you need, the Liberty profile provides the smallest runtime footprint for applications. This dynamic runtime allows the adding of features and updating of configuration parameters without requiring you to restart the server.

Liberty profile: Composable runtime

- Full profile includes everything
- Liberty profile includes only those features you add
 - Improved performance
 - Faster server starts
- **Note:** Problem determination for Liberty is not covered in this course



© Copyright IBM Corporation 2013

Figure 1-30. Liberty profile: Composable runtime

WA5913.0

Notes:

Application servers in the full profile have a JVM shown on the left that includes all services (features) whether the applications require them or not.

Liberty profile server JVMs start only those features that you add to its `server.xml` file. By default, a server contains only the `jsp-2.2` feature to support servlet and JSP applications. You use the feature manager to add the features that you need.

The building blocks shown in the graphic on this slide show some of the Liberty profile features that can be defined for the feature manager. These features include JPA, JSP, servlet, application security, and a remote JMX connector. However, several other features can be configured for a particular server. In addition to the features, you can add HTTP port definitions for the HTTP transport, and application definitions for the application manager.

Intelligent Management (1 of 2)

- Previously the WebSphere Virtual Enterprise product
 - Basically all of the same function as WebSphere Virtual Enterprise
 - No additional license fee required
- Application infrastructure virtualization
 - Run applications on any application server in a common resource pool
 - Adjust resources quickly and seamlessly during peak periods
 - Adjust resources quickly and seamlessly in response to the peaks in unforeseen demand
 - Achieve application response times that meet business policy goals
- Extended manageability
 - Allow seamless application upgrades without experiencing an outage
 - Detect problem scenarios and act before an outage occurs
- Supports the tenets of autonomic computing: self-managing, self-healing, self-configuring, and self-optimizing

© Copyright IBM Corporation 2013

Figure 1-31. Intelligent Management (1 of 2)

WA5913.0

Notes:

There are two major functions of Intelligent Management.

The first is the ability to virtualize your application infrastructure. You might already be using hardware virtualization. Intelligent Management provides another layer of virtualization: virtualizing middleware. Hardware virtualization allows you to share hardware resources, and make efficient use of the resources by using them where they are most needed. Middleware virtualization does the same for your middleware applications. It allows you to share middleware server resources, and make the best use of those resources to serve client requests. Resources can be allocated dynamically as needed to meet unexpected or peak demand.

The second major function of Intelligent Management is extended manageability. As you move to a virtualized environment, you want to make it easier to manage that environment. Although there are many features, there are two key features to consider. The first is the ability to install multiple editions of the same application simultaneously, and easily roll out the new edition or roll back to the previous edition. The second is the ability to watch for health problems with servers, and take steps to fix the server before clients are affected.

Intelligent Management (2 of 2)

- Middleware virtualization
 - Middleware servers are started and stopped dynamically to meet business goals
 - Entire nodes can be brought online and offline to meet business goals – Single cell and multi-cell Performance Management
- Request flow management
 - Prioritizing and routing traffic that is based on business policies
- Enhanced management capabilities
 - Middleware servers are monitored for health problems (health management)
 - Multiple editions of the same application can be managed
 - Real-time and historical details of the performance of the environment can be captured and viewed
 - Servers can be intelligently quiesced
 - Centralized log management for problem determination
- Health management and health policies are covered in a later unit

© Copyright IBM Corporation 2013

Figure 1-32. Intelligent Management (2 of 2)

WA5913.0

Notes:

With middleware virtualization, you use a pool of middleware servers that can be dynamically started and stopped in response to client request flows. You can start more servers as more requests for applications on those servers arrive, or, if requests are not meeting response time goals, you can start more server instances to add more resources for the requests. There is also the ability to add and remove entire nodes in the cell. This ability can bring more resources online, or completely remove servers from the cell if they are no longer needed. Intelligent Management can manage resources within a single cell, and can route requests between multiple cells. It is important to understand that Intelligent Management does not dynamically start and stop applications; it dynamically starts and stops application servers.

Intelligent Management also uses Request Flow Management to optimize the use of resources in your environment. Higher priority requests can be given more resources, and application servers can be protected from overload.

Intelligent Management also provides enhanced management of the middleware environment. Health monitoring can detect problems with servers and take corrective action before clients become aware that there is a problem. Multiple editions of the same application are managed intelligently. The statistics that Intelligent Management uses for decision making and the statistics

about the performance of your environment are logged, and the information can be analyzed to help with setting service policies and with capacity planning. Servers can be put into maintenance mode, so that client traffic can be shifted to other servers, before taking a server offline. There is also the ability to gather logs in a central location, and to dynamically enable trace strings as based on the arrival of particular requests.

1.3. Client request flow

Client request flow



© Copyright IBM Corporation 2013

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

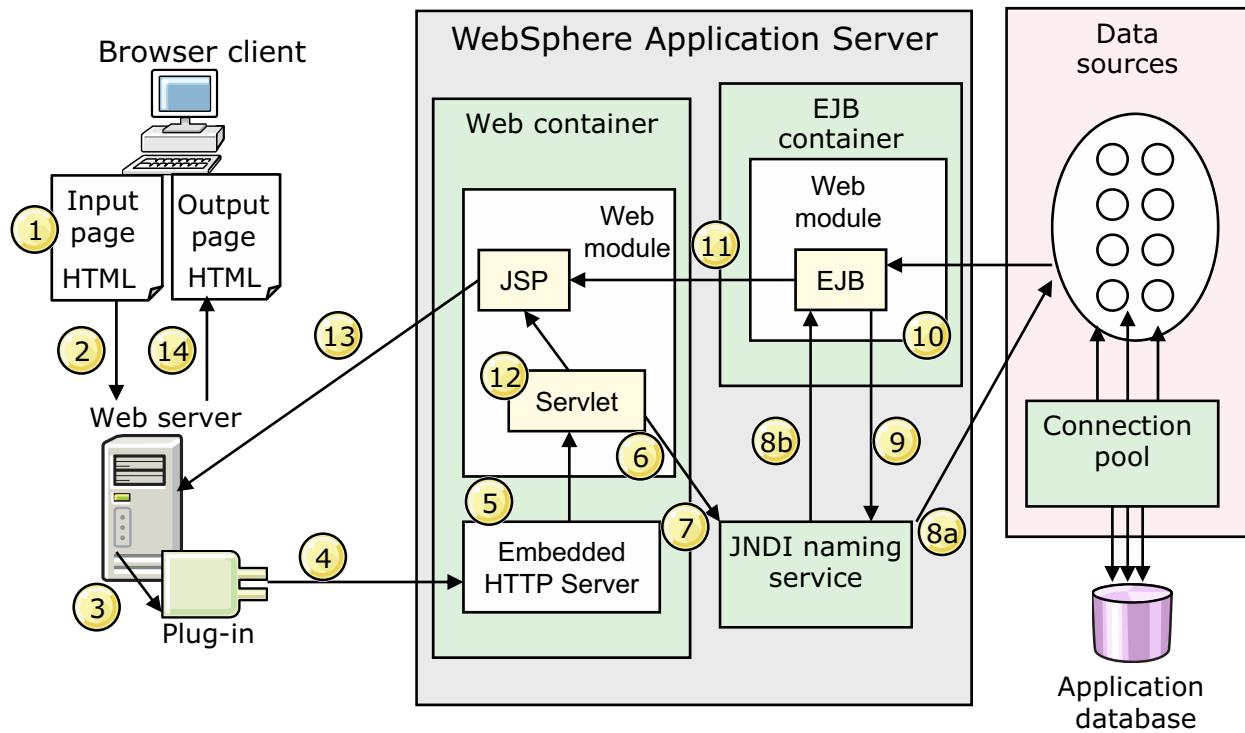
8.0

Figure 1-33. Client request flow

WA5913.0

Notes:

Typical client request application flow



© Copyright IBM Corporation 2013

Figure 1-34. Typical client request application flow

WA5913.0

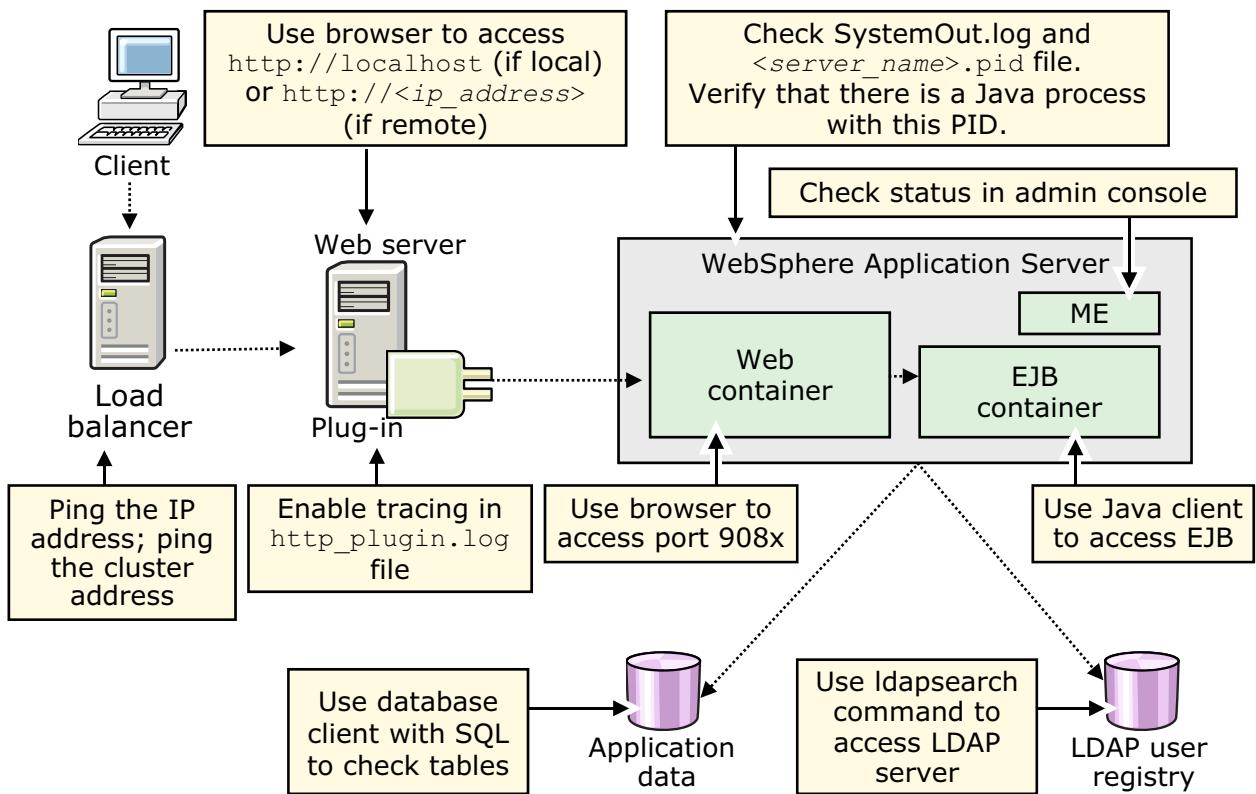
Notes:

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 can 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 correct 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:
 - a. To the corresponding database and gets a connection from its connection pool in the case of a data source
 - b. 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 that is 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.

Are all components in the application flow accessible?



© Copyright IBM Corporation 2013

Figure 1-35. Are all components in the application flow accessible?

WA5913.0

Notes:

This diagram shows various techniques to verify that all components in an application request flow are accessible. When troubleshooting problems reported by clients, one strategy that can be used is to try to reproduce the problem, follow the request flow, and isolate one or more components that might be failing.

Unit summary

Having completed this unit, you should be able to:

- Describe stand-alone server architecture
- Describe Network Deployment (ND) cell architecture
- List and describe the function of IBM products that are involved in implementing stand-alone and distributed architectures
- Identify the components of the application server and describe the services that they provide
- Identify the components of an ND cell and describe the function of each
- Describe the flexible management model
- Describe the various types of clients for the application server
- Describe the flow of an application request
- Describe the flow of security and administration requests
- Identify common troubleshooting points in the end-to-end flow of client requests

© Copyright IBM Corporation 2013

Figure 1-36. Unit summary

WA5913.0

Notes:

Checkpoint questions (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. DB2 database

© Copyright IBM Corporation 2013

Figure 1-37. Checkpoint questions (1 of 2)

WA5913.0

Notes:

Write your answers here:

1.

2.

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

© Copyright IBM Corporation 2013

Figure 1-38. Checkpoint questions (2 of 2)

WA5913.0

Notes:

Write your answers here:

3.

4.



Checkpoint answers

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

© Copyright IBM Corporation 2013

Figure 1-39. Checkpoint answers

WA5913.0

Notes:

Unit 2. Using the IBM Support Assistant Team Server 5.0

What this unit is about

This unit explains how to use the IBM Support Assistant Team Server.

What you should be able to do

After completing this unit, you should be able to:

- Describe the various components of IBM Support Assistant Team Server
- Install, update, and remove problem determination tools
- Manage diagnostic files with cases
- Launch problem determination tools
- View reports that are generated during problem analysis
- Perform automated analysis and review results
- Install the IBM Support Assistant Team Server 5.0

How you will check your progress

- Checkpoint questions
- Lab Exercises

References

Announcing IBM Support Assistant 5 Beta 3

<http://www.ibm.com/support/docview.wss?uid=swg27036157>

Introduction video

<https://www.youtube.com/watch?v=tRlI3YA8nN8>

Support Authority on developerWorks

http://www.ibm.com/developerworks/websphere/techjournal/1208_supauth/1208_supauth.html

Unit objectives

After completing this unit, you should be able to:

- Describe the various components of IBM Support Assistant Team Server
- Install, update, and remove problem determination tools
- Manage diagnostic files with cases
- Launch problem determination tools
- View reports that are generated during problem analysis
- Perform automated analysis and review results
- Install the IBM Support Assistant Team Server 5.0

© Copyright IBM Corporation 2013

Figure 2-1. Unit objectives

WA5913.0

Notes:



Topics

- IBM Support Assistant overview
- IBM Support Assistant Team Server administration
- Case management
- Problem determination tools
- Automated analysis
- Installing IBM Support Assistant 5

© Copyright IBM Corporation 2013

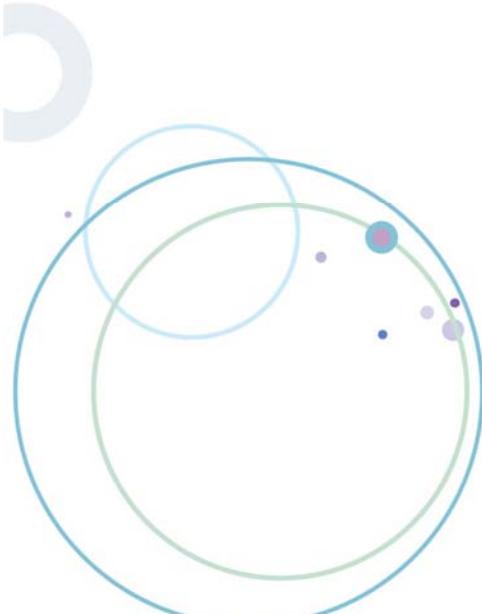
Figure 2-2. Topics

WA5913.0

Notes:

2.1. IBM Support Assistant overview

IBM Support Assistant overview



© Copyright IBM Corporation 2013

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

8.0

Figure 2-3. IBM Support Assistant overview

WA5913.0

Notes:

What is the IBM Support Assistant Team Server 5.0?

- Free self-help problem determination application
 - Cloud based
 - Multiple installation options
- Use to organize diagnostic files into cases
 - Store diagnostic files by problem incident or other categories
 - Simplifies collaboration
- Provides a growing collection of problem determination tools
 - Report generators
 - Interactive web-based tools
 - Desktop tools
- Provides automated analysis of diagnostic files
- Download IBM Support Assistant Team Server 5.0

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

© Copyright IBM Corporation 2013

Figure 2-4. What is the IBM Support Assistant Team Server 5.0?

WA5913.0

Notes:

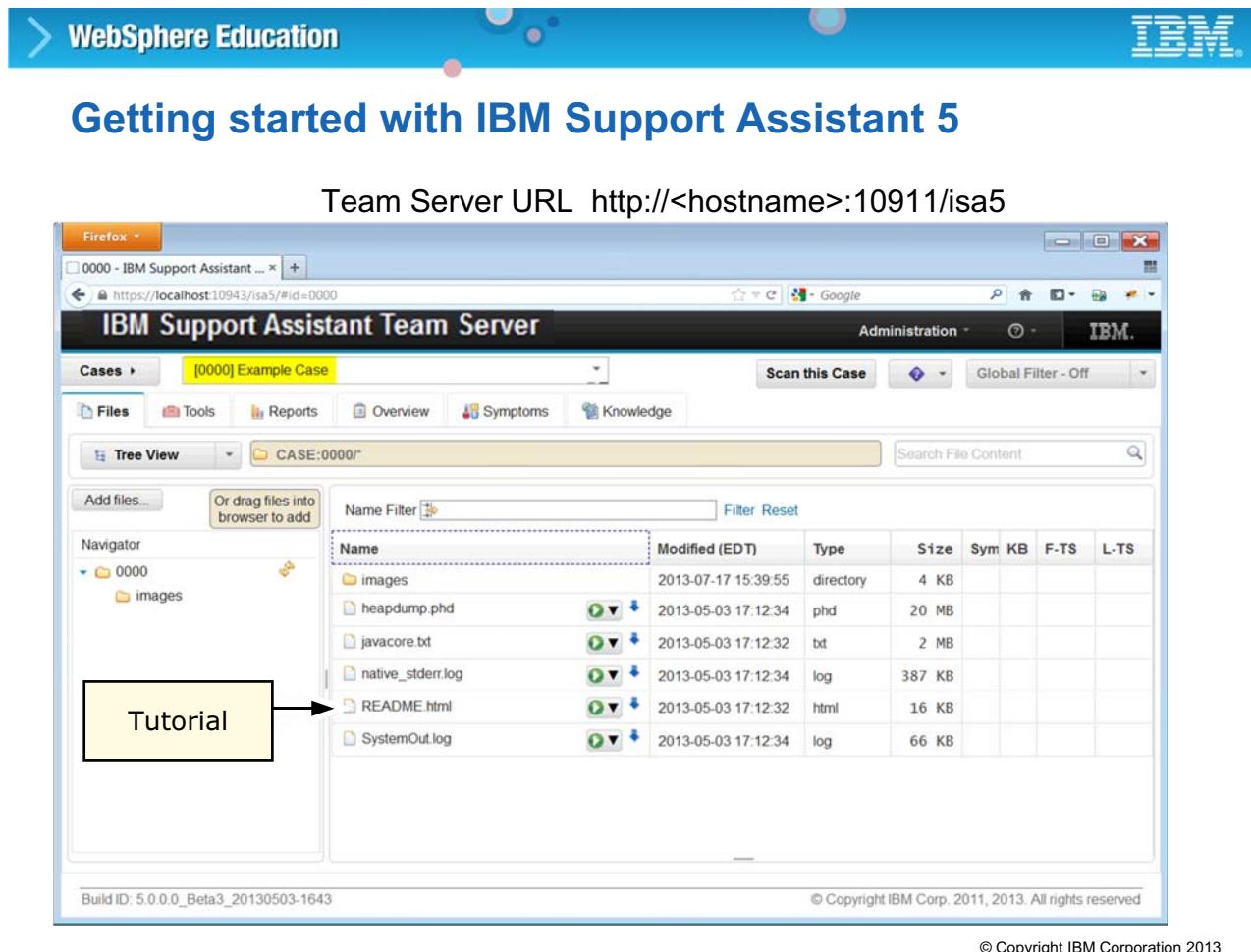


Figure 2-5. Getting started with IBM Support Assistant 5

WA5913.0

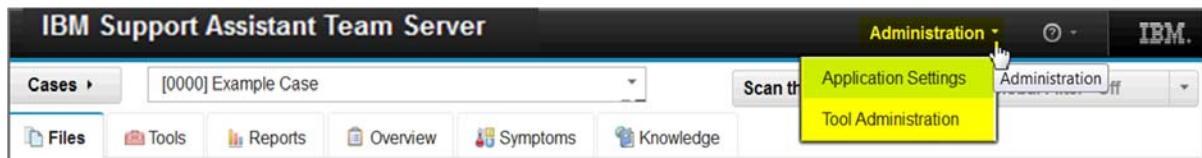
Notes:

If you are new to ISA 5 then you should start with the tutorial in the example case. Start the ISA server and then open a browser and point it to <http://<hostname>:10911/isa5>. After ISA loads, choose the “Example Case” from the “Cases” dropdown under the black banner. The files for the case will load. Double-click README.html to open the tutorial. It will introduce you to ISA's features by guiding you through a troubleshooting workflow.

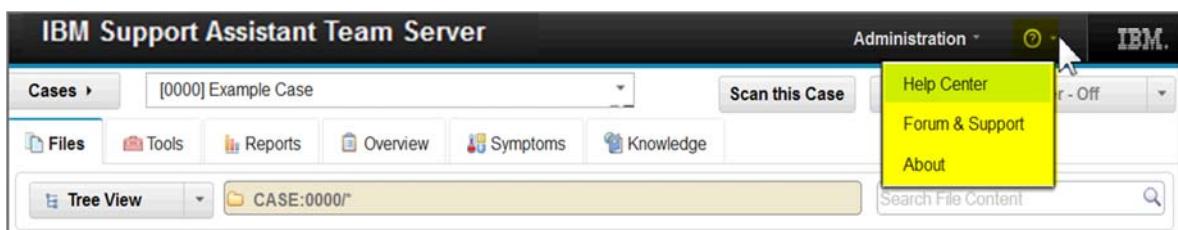


Banner: Features

- Administration (secured with user ID and password)



- Help



- Language menu (not shown)

© Copyright IBM Corporation 2013

Figure 2-6. Banner: Features

WA5913.0

Notes:

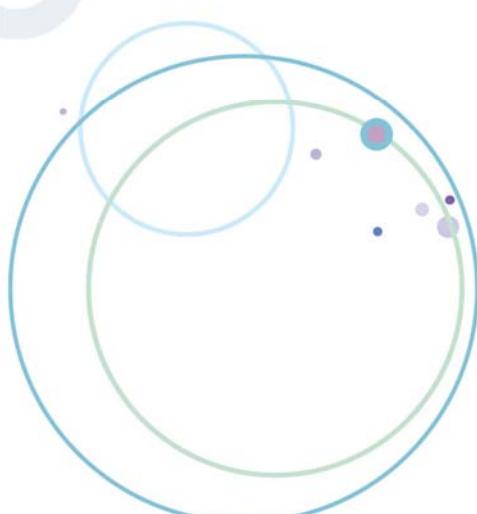
There are two menus in the banner: one for documentation and support and the other for administrative tasks.

Under the Administration menu you can choose to configure settings for ISA or manage your problem determination tool installations.

Under the Help menu you can reach the documentation for ISA and all problem determination tools that are currently installed. You can also reach the ISA forum, which is the official channel for asking questions and reporting bugs.

2.2. IBM Support Assistant Team Server administration

IBM Support Assistant Team Server administration



© Copyright IBM Corporation 2013

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

8.0

Figure 2-7. IBM Support Assistant Team Server administration

WA5913.0

Notes:

The screenshot shows the 'IBM Support Assistant Team Server' administration console. The top navigation bar includes 'Administration', a help icon, and the 'IBM' logo. Below the bar, the title 'Administration: Application Settings' is displayed. The main content area has tabs for 'Application Settings' (which is selected and highlighted in yellow) and 'Tools Administration'. On the left, there's a sidebar with a 'Usage Statistics' section. The right side contains a 'Usage Statistics' panel with descriptive text about usage data collection, terms of use, and an enable checkbox. At the bottom right of the panel is an 'Apply' button.

© Copyright IBM Corporation 2013

Figure 2-8. Administration: Application Settings

WA5913.0

Notes:

In Beta 3 and GA the only application settings are for configuring the collection of usage data. In Beta 3 you must opt in to participate. In the GA opt-in is the default setting so you must choose to opt out if you do not want to participate.

Administration: Tools Administration

IBM Support Assistant Team Server

Administration Console

Tools Administration

Memory Analyzer [Report] Version 1.2.0.201208221220-FP1

Uninstall

**Install
Update
Uninstall actions**

No Screenshot Available

Memory Analyzer is a feature-rich Java heap analyzer that helps you find memory leaks and reduce memory consumption.

This tool is provided in three versions:

- as a report generating version that reads the dump, builds some index files and generates some zipped HTML reports
- as an inter... running on ...
- as an interac...

(more)

Tags: Report Generator Tool | Problem Area: Memory | Problem Area: Java | Supported | Installed | Family: IBM Monitoring and Diagnostic Tools for Java

Restrictions: If analyzing a large dump, this tool should be run on a 64-bit machine.

File Types Hint: The file or pair of files from one line of the following list should be selected as input to the tool:

© Copyright IBM Corporation 2013

Figure 2-9. Administration: Tools Administration

WA5913.0

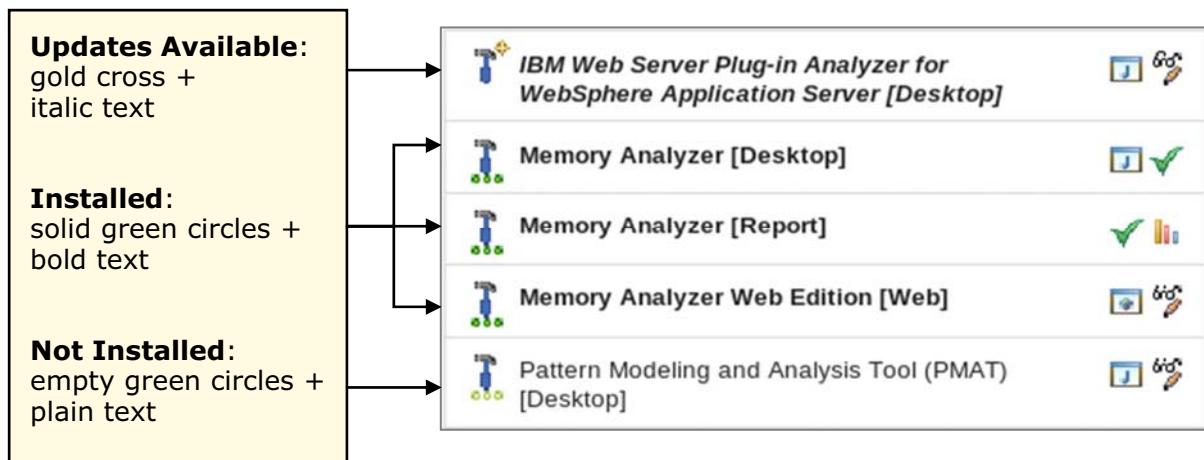
Notes:

The Tools Administration tab shows you a list of all problem determination tools that are available for IBM Support Assistant 5. This tab lets you install, update, and uninstall problem determination tools. The column on the left shows you the complete list of tools. Under the tool title above the details pane you'll find buttons for installing, updating, or uninstalling tools depending on the current installation state.

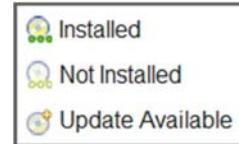
The next slides show you the various ways to recognize the installation state.

Administration: Tool installation state indicators

- Visual cues to tell you whether a tool is installed, uninstalled, or installed with an update available



- In addition, a set of icons show state in the tags list and in the tags that are shown on the details pane for each tool



© Copyright IBM Corporation 2013

Figure 2-10. Administration: Tool installation state indicators

WA5913.0

Notes:



Administration: Tool catalog details pane

- Text describing details about the tool
- Latest news and release history
- Tags
- Restrictions
- File Types Hint (what types of diagnostic files does the tool analyze)

What's New:

Version 2.7.0.201305230714 / V2.7.0.1

- Generates all reports by default. Generation of table can be disabled.
- Creates parent report for navigating through the collection of generated reports.

Tags: Report Generator Tool | Problem Area: Memory | Problem Area: Java |
✓ Supported | Installed | Family: IBM Monitoring and Diagnostic Tools for Java |
Problem Area: Performance |

Restrictions:

File Types Hint: Verbose GC logs

© Copyright IBM Corporation 2013

Figure 2-11. Administration: Tool catalog details pane

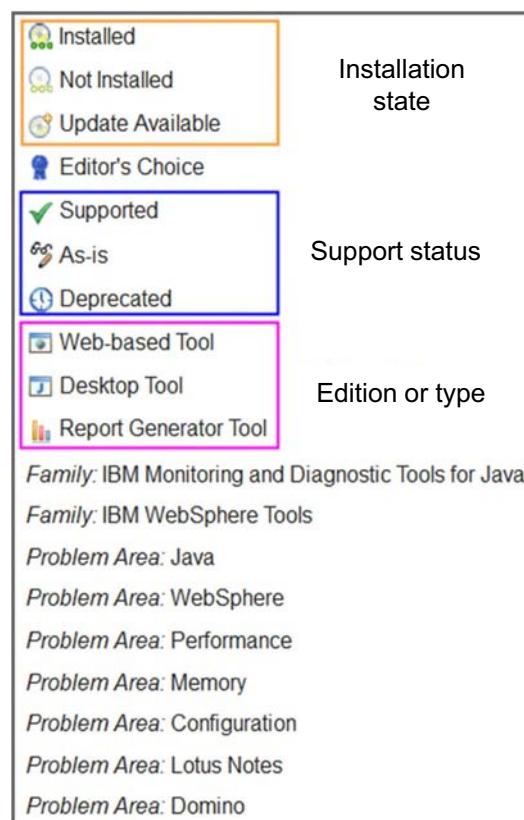
WA5913.0

Notes:



Administration: Tool tags

- Installation state (catalog only)
- Editor's choice
 - Best among other tools of same type
- Support status
- Edition or type
 - Web
 - Desktop
 - Report Generator
- Family
- Problem area



© Copyright IBM Corporation 2013

Figure 2-12. Administration: Tool tags

WA5913.0

Notes:



Administration: Tool installation

1. Select a tool from the list
2. Click the installation action button beneath the tool name
3. Provide your IBM ID and password
4. Check the box to accept the license agreement
5. Click **OK** to begin the action

The screenshot shows a 'Install Tool' dialog box. On the left, there is a sidebar with the text 'HeapAnalyzer [Desktop] Version 4.4.7.0' and a yellow 'Install' button. The main area contains the following fields:

- 'License Terms' link (highlighted with a yellow box and an arrow pointing to it)
- A checked checkbox labeled 'I accept the terms in the license agreement'
- 'IBM ID' field with value 'myIBMid'
- 'Password' field with masked value '*****'
- 'Continue' and 'Cancel' buttons at the bottom

© Copyright IBM Corporation 2013

Figure 2-13. Administration: Tool installation

WA5913.0

Notes:

2.3. Case management

Case management



© Copyright IBM Corporation 2013

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

8.0

Figure 2-14. Case management

WA5913.0

Notes:



Case management

- Manage problem incidents with cases
 - Containers for logically grouping files, reports, and other information
 - Save notes about each case
 - Upload diagnostic files to store in cases

The screenshot shows the 'Case Management' interface of the IBM Support Assistant Team Server. At the top, there's a header bar with the IBM logo and the title 'IBM Support Assistant Team Server'. Below the header, on the left, is a sidebar with tabs for 'Cases', 'Files', 'Tools', 'Reports', and 'Logs'. The 'Cases' tab is currently selected. The main content area is titled 'Case Management' and contains a table with one row. The table has columns for 'Case ID' and 'Summary'. The 'Case ID' column shows '0000' and the 'Summary' column shows 'Example Case'. Below the table, there's a form with fields for 'Case ID' (0000) and 'Summary' (Example Case). Underneath the form is a 'Description:' section with some text. On the right side of the interface, there's a vertical sidebar with sections for 'Cases' and 'Knowledge'.

© Copyright IBM Corporation 2013

Figure 2-15. Case management

WA5913.0

Notes:

Before you begin troubleshooting a problem you should create a case to manage related diagnostic files. Cases let you organize diagnostic artifacts by problem incident, failing system, time, or any other logical categories that you like. When you create a case you assign a summary and you can optionally add details.

Click the “Cases” button above the Files tab to open “Case Management” slider where you can add, edit, and delete cases.

After you have a case you're ready to start adding files to it.



Files tab: Add files to a case

- Click the **Add files** button
- Browse file system or drag files into the browser

The screenshot shows the 'IBM Support Assistant Team Server' interface in a Firefox browser. The title bar says 'IBM Support Assistant Team Server'. The main area has tabs for 'Cases' (selected), 'Tools', 'Reports', 'Overview', 'Symptoms', and 'Knowledge'. Below these is a 'Tree View' section with 'CASE:0003/' selected. On the left is a 'Navigator' pane showing a folder structure with '0003' expanded. The right pane displays a table of files in 'CASE:0003/'. The table has columns: Name, Modified (EDT), Type, Size, Sym, KB, F-TS, and L-TS. One file, 'SystemOut.log', is listed with a size of 400 KB. At the top of the right pane, there are buttons for 'Add files...', 'Or drag files into browser to add', 'Name Filter', 'Filter', and 'Reset'. A search bar says 'Search File Content'.

Name	Modified (EDT)	Type	Size	Sym	KB	F-TS	L-TS
SystemOut.log	2013-07-31 17:13:30	log	400 KB				

© Copyright IBM Corporation 2013

Figure 2-16. Files tab: Add files to a case

WA5913.0

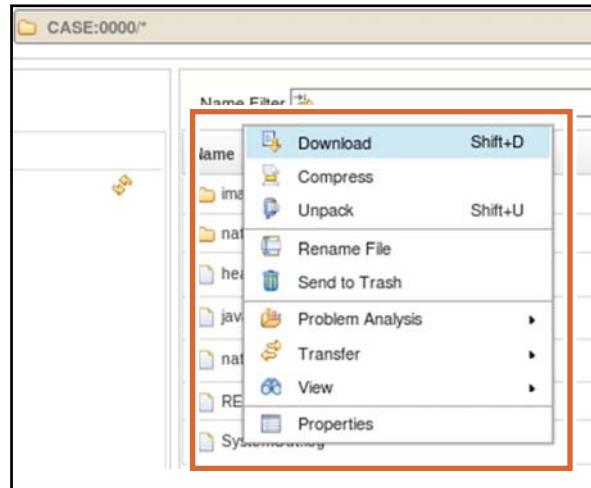
Notes:

There are two ways to add files to a case: the “Add files...” button and “drag and drop”. Clicking “Add files...” opens the browser’s file selection window where you can select one or more files. After you select your files a progress bar will appear and when the upload is complete you’ll see a yellow notification message at the top of the ISA UI.

“Drag and drop” lets you grab files from your OS desktop and drop them into the selected case. Just as with “Add files...” you’ll see a progress bar and a notification message when the upload is complete.

Files tab: Operating on case files

- Operate on files that are stored in cases
- Right-click a file to open the Files menu
 - Compress and extract files with various compression types
 - Move files between cases
 - Download case files to your workstation
 - View contents of text files in the browser
 - Delete case files
 - Analyze files with problem determination tools



© Copyright IBM Corporation 2013

Figure 2-17. Files tab: Operating on case files

WA5913.0

Notes:

2.4. Problem determination tools

Problem determination tools



© Copyright IBM Corporation 2013

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

8.0

Figure 2-18. Problem determination tools

WA5913.0

Notes:

Problem determination tools: Overview

- Growing collection of problem determination tools for various software products
 - Java
 - WebSphere Application Server
 - WebSphere Portal
- Multiple editions (types or styles)
 - Report generators
 - Interactive web tools
 - Desktop tools
- Tools tab to view the contents of your toolbox of installed tools
- Reports tab to easily find analysis reports
- Tools Administration tab to manage your tool installations

© Copyright IBM Corporation 2013

Figure 2-19. Problem determination tools: Overview

WA5913.0

Notes:

Editor's choice and recommended tools

- Memory Analyzer
 - Java heap analyzer that helps you find memory leaks and reduce memory consumption (report, web, desktop)
- Thread and Monitor Dump Analyzer (TMDA)
 - Compares each thread dump and monitor dump and automatically detects hangs, resource contention, Java monitor ownership directional graph structure, and deadlocks.
- Garbage Collection and Memory Visualizer (GCMV)
 - Provides graphical display of a wide range of verbose GC data values together with tuning recommendations and detection of problems such as memory leaks
- Health Center
 - A lightweight tool that monitors active IBM Virtual Machines for Java with minimal performance overhead and provides live tuning recommendations and observations.
- Interactive Diagnostic Data Explorer (IDDE)
 - Use to more easily explore and examine system dump files that are produced by the JVM.

© Copyright IBM Corporation 2013

Figure 2-20. Editor's choice and recommended tools

WA5913.0

Notes:

Problem determination tools: Launching tools

- Three methods for launching tools
 - Files tab Quick Tool Launcher
 - Files tab menu
 - Tools tab Launch button

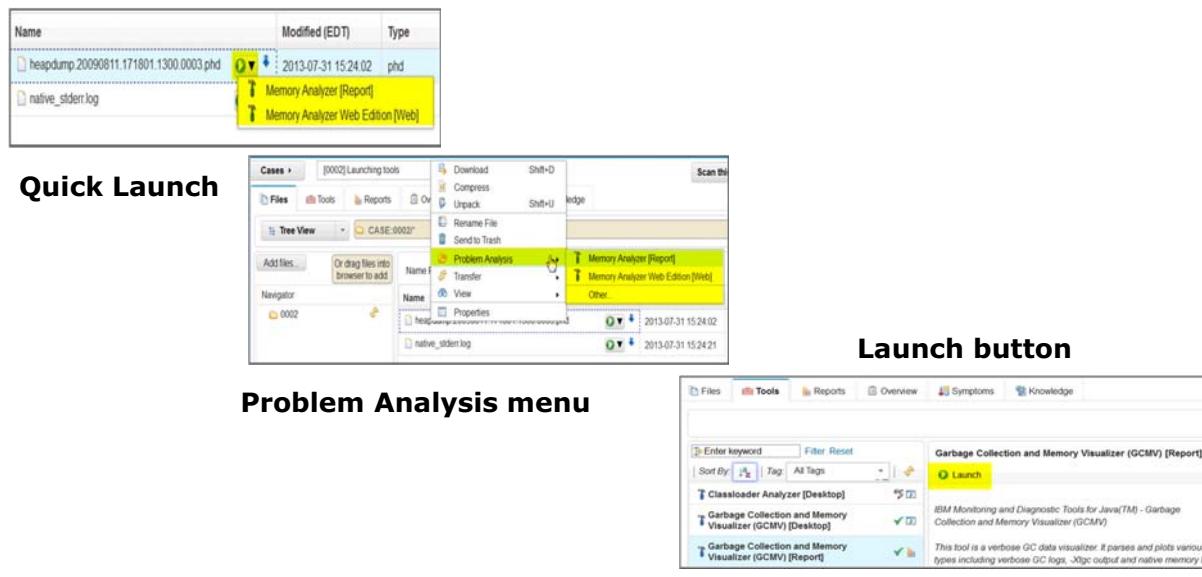


Figure 2-21. Problem determination tools: Launching tools

WA5913.0

Notes:

There are three methods for launching tools. On the Files tab you can choose the QuickLaunch icon next to a file name or right click on one or more files and choose a tool from the "Problem Analysis" item of the popup menu. The lists are filtered based on the type of file selected to show you which tools might be applicable. If the tool you need isn't listed then you can select "Problem Analysis > Other..." to choose from the complete list of installed tools.

You can also launch tools from the Tools tab. Under the title of each tool on its details pane is a Launch button. When you click launch for report and web tools you'll see a browse button on the input dialog to let you choose the case files for analysis. Note that desktop tools are available only from the Tools tab.



Problem determination tools: Report generators

- Accept input files and parameters through the browser
- Produce static reports accessible from the Reports tab

Launch dialog

Report

Garbage Collection and Memory Visualizer (GCMV) [Report]

Memory

Properties of the heap.
Useful for diagnosing memory leaks or suboptimal heap sizing.

[Report](#)
[Line plot](#)

Performance

Garbage collection pauses.
Useful for investigating performance problems.

[Report](#)
[Line plot](#)

© Copyright IBM Corporation 2013

Figure 2-22. Problem determination tools: Report generators

WA5913.0

Notes:

Report generators accept one or more files or folders as input along with any additional parameters the tool needs to run. It performs its work and when it's finished you can view the results on the Reports tab.

Reports are organized by case (you only see reports for the case that is currently open).

The status icon to the left of the report name in the list of reports indicates success (green check mark), failure (red X), or in progress (hourglass). Other information in the report list includes the time the report was run and the input file(s). When you hover over an item a tooltip opens showing input files and input parameters.

You can filter the list using using keywords and you can sort it alphabetically or by time. The default sorting is by time with the most recent report at the top of the list.

The larger right pane shows the report. The icons beneath the report title and above the report let you take additional actions.

- The left icon lets you open the report in a new browser tab.
- The middle icon switches to the Files tab and navigates to the folder containing the input file(s).

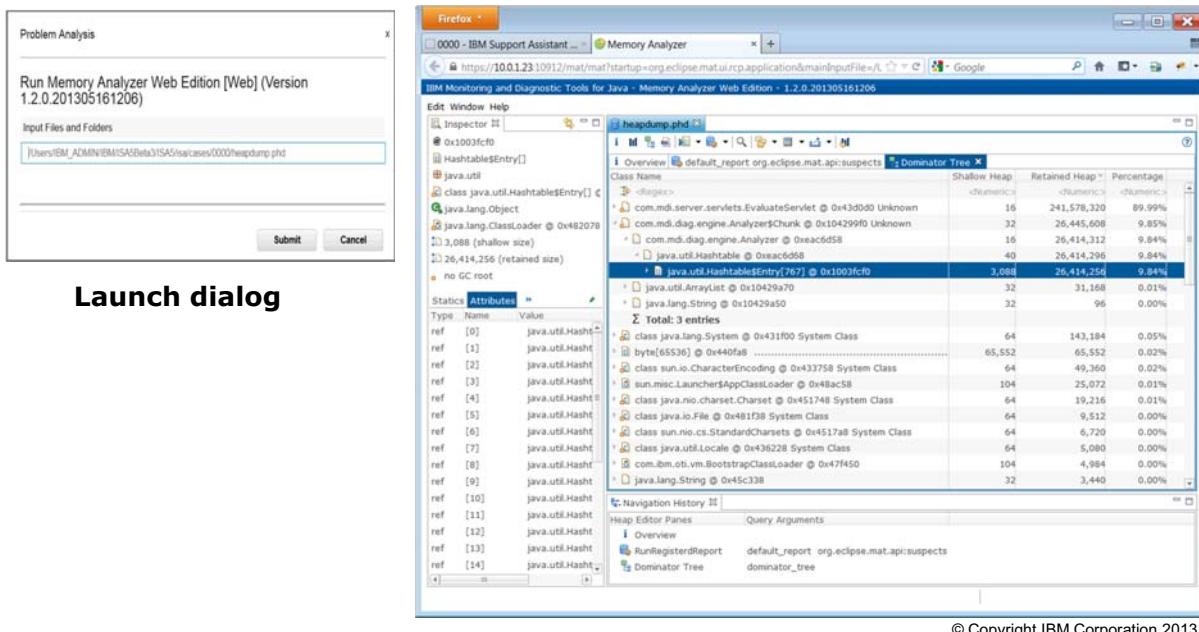
- The right icon (graph in a folder) switches to the Files tab and navigates to the folder containing the report files and logs.



Problem determination tools: Web tools

- Select input files and input parameters through a launch dialog
- Interact through a session with the web tool in a new tab

Web tool session



© Copyright IBM Corporation 2013

Figure 2-23. Problem determination tools: Web tools

WA5913.0

Notes:

Web tools open in a separate browser tab or window. They are interactive like desktop tools, but the work is performed on the ISA server; the tool does not need to be installed on your workstation and using it does not consume any workstation resources.



Problem determination tools: Desktop tools

- Select a desktop tool from the Tools tab
- Tool downloads (via Java Web Start) and runs on your local workstation

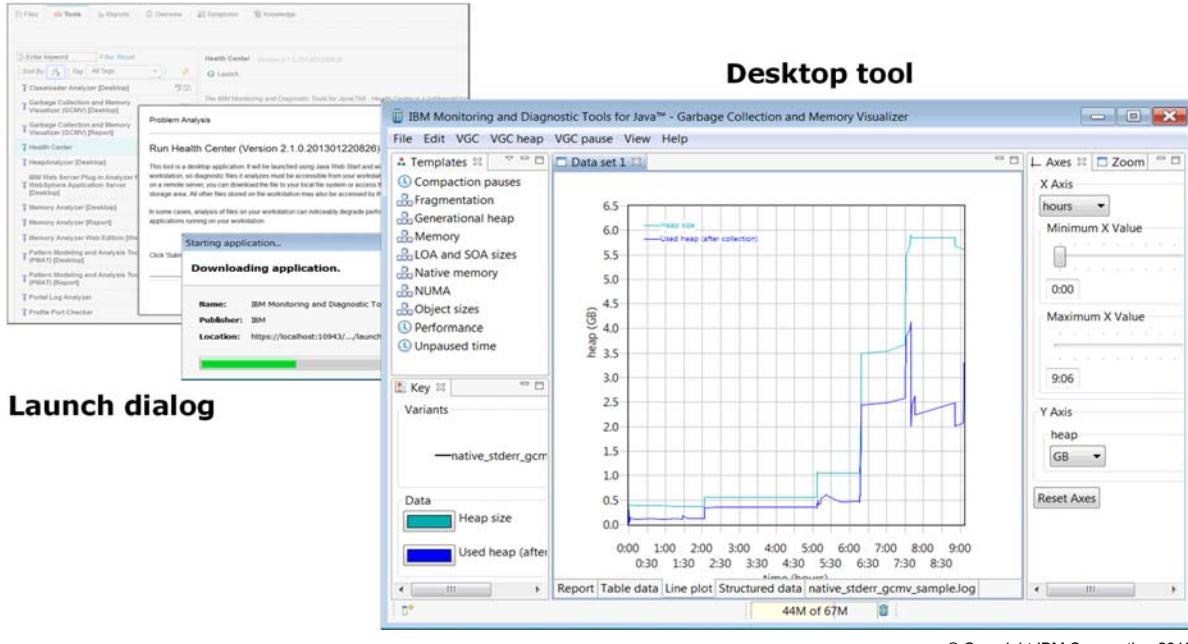


Figure 2-24. Problem determination tools: Desktop tools

WA5913.0

Notes:

Desktop tools run on your workstation's desktop. This is the only type of tool offered with IBM Support Assistant 4.

ISA uses Java Web Start technology to support desktop tools. When you choose to launch a desktop tool the browser will either ask you what to do when you click "Submit" on the confirmation dialog or it will automatically start downloading the tool and then launch it. Note that this does not require a Java plug-in or any other type of browser extension, it uses the built-in download capability of the browser.

Desktop tools allow you only to work with diagnostic files stored locally on your workstation or on a mounted file system. If the cases file system is not mounted for your workstation, then you would need to first download case files to your workstation to analyze them.



Tools tab: Overview

- Catalog of all tools that are installed (Toolbox)
- Contains detailed information about each tool
- Includes the ability to launch tools

The screenshot shows the 'Tools' tab in the WebSphere Application Server V8.5.5 Problem Determination interface. The left pane lists various tools such as Classloader Analyzer, Garbage Collection and Memory Visualizer, Health Center, HeapAnalyzer, IBM Web Server Plug-in Analyzer, Memory Analyzer, Memory Analysis Web Edition, Pattern Modeling and Analysis Tool (PMAT), Portal Log Analyzer, Profile Port Checker, Thread and Monitor Dump Analyzer (TMDA), Trace and Request Analyzer, and WebSphere Application Server Configuration Visualizer. The right pane provides detailed information for the 'Garbage Collection and Memory Visualizer (GCMV) [Report]' tool, including its description, tags (Report Generator Tool, Supported, Family: IBM Monitoring and Diagnostic Tools for Java, Problem Area: Java, Performance, Memory), restrictions, execution history (status: 2013-08-01 08:20:26, input file: /Users/IBM_ADMIN/IBM/ISAS5Beta3/AS5/isa/cases/0000/native_stderr.log, additional parameters: GenerateTableData: false), and a note that no screenshot is available.

© Copyright IBM Corporation 2013

Figure 2-25. Tools tab: Overview

WA5913.0

Notes:

The Tools tab contains most of what you need to know about each problem determination tool. Additional documentation is available from the “Tool Help” link above each catalog entry.

The Tools is a catalog of all problem determination tools that are installed for this ISA installation, so it may be a subset of all available tools. An administrator can install, update, and remove tools. Because this is a catalog of installed tools it's sometimes referred to the toolbox.

The column on the left is the list of tools. At the top of the column there are features for filtering and sorting the list. You can sort alphabetically and you can filter using text strings and various tags such as tool type, support status, and types of problems applicable to a tool.

The list itself includes the name of the tool, a hover tooltip with summary information, and icons that tell what kind of tool it is, whether or not it's supported, whether or not it's deprecated, and whether or not it's identified as an “editor's choice”.

The information panel on the right provides more information about each tool. There's also a search box above the details pane that lets you search through the documentation for installed problem determination tools. Additional information about the Tools tab will be covered on additional slides.



Tools tab: List of installed tools

Icons Legend

	Editor's Choice
	Supported
	As-is
	Deprecated
	Web-based Tool
	Desktop Tool
	Report Generator Tool

Filter and sort

Tool Name	Tags	Action Buttons
Classloader Analyzer [Desktop]		
Garbage Collection and Memory Visualizer (GCMV) [Desktop]	✓	
Garbage Collection and Memory Visualizer (GCMV) [Report]	✓	
Health Center	✓	
HeapAnalyzer [Desktop]		

Refresh list

Tool Name	Tags	Action Buttons
Classloader Analyzer [Desktop]		
Garbage Collection and Memory Visualizer (GCMV) [Desktop]	✓	
Garbage Collection and Memory Visualizer (GCMV) [Report]	✓	
Health Center	✓	
HeapAnalyzer [Desktop]		

© Copyright IBM Corporation 2013

Figure 2-26. Tools tab: List of installed tools

WA5913.0

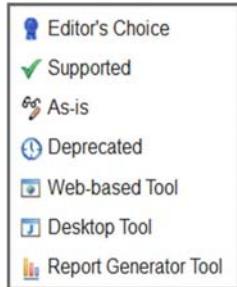
Notes:

The icons legend shows the icons for various tags in the Tags filter. You can filter the list of tools by keyword or by tags and you can sort the list alphabetically. Click the Refresh icon to reload the list of tools in your browser.

 WebSphere Education 

Tools tab: Details pane

Icons Legend



Details pane

Garbage Collection and Memory Visualizer (GCMV) [Report] Version 2.7.0.201305230714

IBM Monitoring and Diagnostic Tools for Java(TM) - Garbage Collection and Memory Visualizer (GCMV)

This tool is a verbose GC data visualizer. It parses and plots various log types including verbose GC logs, -Xtgc output and native memory logs (output from ps, svmon and perfmon). It provides graphical display of a wide range of verbose GC data values together with tuning recommendations and detection of problems such as memory leaks. You can select and parse multipl...

[\(more\)](#)

Tags:  Report Generator Tool |  Supported | **Family:** IBM Monitoring and Diagnostic Tools for Java | **Problem Area:** Java | **Problem Area:** Performance | **Problem Area:** Memory |

Restrictions:

File Types Hint: Verbose GC logs

Execution History/Status

Status	Start Time	Input Files	Additional Input Parameters	Report Details
	2013-08-01 08:20:26	/Users/IBM_ADMIN/IBM/ISA5Beta3/ISA5/isa/cases/0000/native_stderr.log	generateTableData: false	View report

© Copyright IBM Corporation 2013

Figure 2-27. Tools tab: Details pane

WA5913.0

Notes:

The details pane on the Tools tab is very similar to the details you see on the Tools Administration tab. The two primary differences are;

- On the Tools tab there's a Launch button under the title of each tool. On the administration tab there are buttons for installing, updating, or uninstalling tools.
- At the bottom of the details pane on the Tools tab is a log of each execution of this tool for the selected case. It shows success/failure, when the tool was run, its input files and input parameters, and a link to the report.

Reports tab

The screenshot shows the WebSphere Education interface with the Reports tab selected. On the left, a list of reports is shown with an annotation pointing to it labeled "Report list". On the right, detailed content for a specific report is displayed with an annotation pointing to it labeled "Report content". Below this, another section shows sorting functionality with an annotation pointing to it labeled "Sort by time".

Report list

Report content →

Sort by time

Garbage Collection and Memory Visualizer (GCMV) [Report]

Tuning recommendation Tuning recommendation

Version Your application appears to be leaking memory. This is indicated by the used heap increasing at a greater rate than the application workload (measured by the amount of data freed). To investigate further see [Diagnostics Guide](#).

VM settings

Summary

Used heap (after collection) At one point 30336 objects were queued for finalization. Using finalizers is not recommended as it can slow garbage collection and cause wasted space in the heap. Consider reviewing your application for occurrences of the finalize() method. You can use the ISA Tool Add-on, IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer to list objects that are only retained through finalizers.

JVM restarts

Heap size Your application is allocating many large objects, which affects performance. Consider increasing -Xloaminimum and -Xloainitital.

Garbage collection Garbage collection is causing some large pauses. The largest pause was 12215 ms. This may affect application responsiveness. If responsiveness is a concern then a switch of policy or reduction in heap size may be helpful.

Garbage Collection and Memory Visualizer (GCMV) [Report]

1. Open in a new browser
2. Open input folder
3. Open report folder

© Copyright IBM Corporation 2013

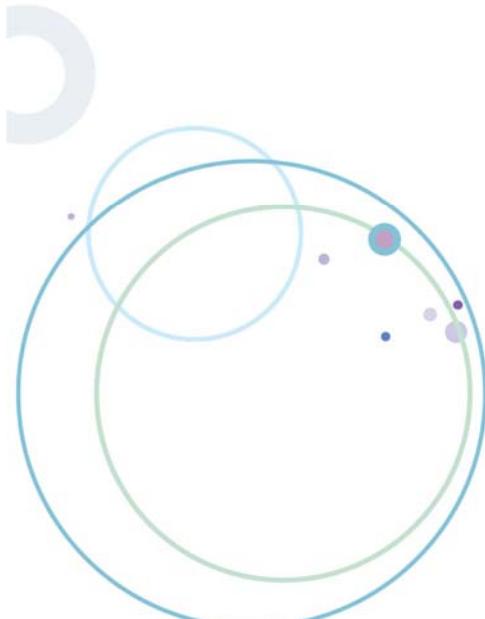
Figure 2-28. Reports tab

WA5913.0

Notes:

2.5. Automated analysis

Automated analysis



© Copyright IBM Corporation 2013

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

8.0

Figure 2-29. Automated analysis

WA5913.0

Notes:



Automated analysis overview

- Automatically analyze case files
 - Runs several built-in problem determination tools to produce reports
 - Scans files to look for known problem symptoms and possible solutions

The screenshot shows the IBM Support Assistant (BETA) interface. At the top, there's a navigation bar with 'IBM Support Assistant (BETA)', 'Administration', and the IBM logo. Below it is a toolbar with 'Cases >', a dropdown for 'Example Case', and buttons for 'Scan this Case', 'Global Filter - Off', and other functions. The main area has tabs for 'Files', 'Tools', and 'Reports', with 'Overview' currently selected. A 'Tree View' sidebar shows a directory structure with '0000' expanded, showing 'images', 'pdtools', and 'pdttools'. The central part displays a table of files from 'CASE:0000/':

Name	Modified (EDT)	Type	Size	Sym	KB	F-TS	L-TS
images	2013-07-17 15:39:55	directory	4 KB				
pdtools	2013-07-29 13:34:33	directory	0 B				
headdump.phd	2013-05-03 17:12:34	jvmHeapDur	20 MB	1	10		
javacore.txt	2013-05-03 17:12:32	jvmJavacore	2 MB	0	0	03/22/11 22:29:47:000	03/22/11 22:29:47:000
native_stderr.log	2013-05-03 17:12:34	logfile.was.s	387 KB	0	0	08/20/12 09:45:02:000	08/21/12 10:47:36:000
README.html	2013-05-03 17:12:32	html	16 KB	0	0		
SystemOut.log	2013-05-03 17:12:34	logfile.was.s	66 KB	12	89	06/08/10 13:54:52:500	06/08/10 14:03:00:812

© Copyright IBM Corporation 2013

Figure 2-30. Automated analysis overview

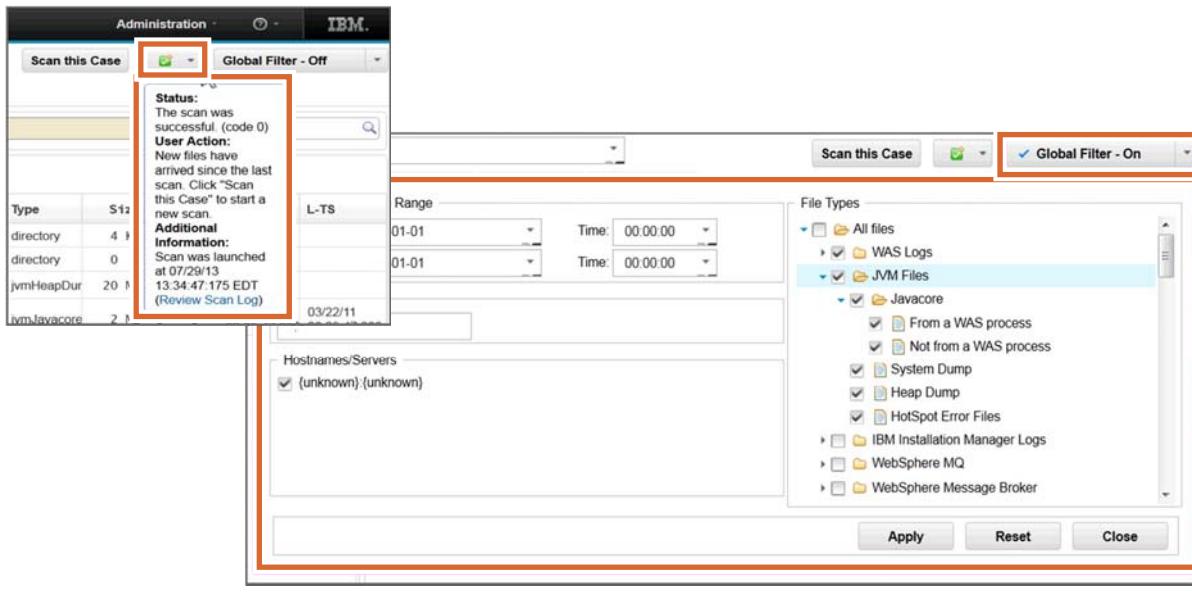
WA5913.0

Notes:



Automated analysis: Perform a scan

- Click **Scan this Case** to start an automated analysis.
- Click the scan status widget for information about scans for this case
- Enable the global filter to filter the contents of the Files tab



© Copyright IBM Corporation 2013

Figure 2-31. Automated analysis: Perform a scan

WA5913.0

Notes:

Click “Scan this Case” to start a new scan or a re-scan. The scan will examine every file in the case including the contents of archives (for example, zip, jar, and so forth.).

The scan status widget shows you information about the state of the scan—has a scan been performed? Was the last scan successful? When was the last scan?

The global filter lets you set criteria for which scanned files appear under the Files tab. For example, time ranges within the files, file names, and file types.



Automated analysis: Filtered Files tab (Flat view)

- Number of symptoms in each file
- Number of knowledge base hits
- First and last time stamp per file segment
- File modification time
- File type

Name	Symptom Occurrences	Knowledge Base Matches	First Timestamp	Last Timestamp	Size	Last Modified	Type
\heapdump.phd	1	10			20 MB	2013-05-03 17:12:34	jvmHeapDump
\native_stderr.log	0	0	08/20/12 09:45:02:000	08/21/12 10:47:36:000	387 KB	2013-05-03 17:12:34	logfile.was.server.native_stderr.ver1
native_stderr.log Segment#1			08/20/12 09:45:02:000	08/21/12 06:19:18:000	170 KB	2013-05-03 17:12:34	logfile.was.server.native_stderr.ver1
native_stderr.log Segment#2			08/21/12 08:01:40:000	08/21/12 10:47:36:000	217 KB	2013-05-03 17:12:34	logfile.was.server.native_stderr.ver1

native_stderr.log Segment#1

General Information

Last modified: 2013-05-03 17:12:34
 First time stamp: 08/20/12 09:45:02:000
 Last time stamp: 08/21/12 06:19:18:000
 Size: 170 KB
 File Type (logical): logfile.was.server.native_stderr.verboseGC
 Knowledge base matches: -1
 Symptom occurrences: -1

© Copyright IBM Corporation 2013

Figure 2-32. Automated analysis: Filtered Files tab (Flat view)

WA5913.0

Notes:

When the global filter is on and the “Flat View” of the Files tab is selected, the list of files is limited by the filter criteria.

Files that contain multiple events such as restarts are segmented into those time frames.

The “Symptom Occurrences” column tells you how many defined symptoms were found and the “Knowledge Base Matches” tells you how many possible solutions (For example, Technotes, APARs) were found for that symptom.

The Type column shows the type of the diagnostic file. The types are determined by peeking inside the file to determine its type, regardless of its file name extension. For example, this lets you tell whether a .log file is from WebSphere or DB2.

When you click a row in the table of files, details about that file are displayed below the table.



Automated analysis: Overview tab

- Summary of what was found during the scan
- Indication of how much of the case contents were recognized
- Generate a printable HTML report

The screenshot shows the 'Overview' tab selected in the top navigation bar. A 'Scan Coverage' progress bar is visible. On the right, there is a 'Printable version' link. The main content area displays general information from a scan, including the most recent scan details (date: Mon Jul 29 13:34:47 EDT 2013), file and symptom counts, and a list of discovered components and tools. A sidebar on the left lists various sections such as Product Versions, JVM Versions, WebSphere Versions, OS Versions, iFix Information, Database Information, Server Names, Server Instances, Host Names, Trace Specifications, Notable Collections and Files, and Special Tool Reports.

© Copyright IBM Corporation 2013

Figure 2-33. Automated analysis: Overview tab

WA5913.0

Notes:

The Overview tab summarizes what it found out about the software and runtime environment for the files contained in the case. It also contains a section with links to problem determination tool reports that were generated during the scan.

The “scan coverage” bar shows how much data within the case was recognized based on the knowledge base that was available.

The “Printable Version” link creates a navigable HTML representation that can be printed.



Automated analysis: Symptoms tab

- Lists each known symptom that is found during the scan
- Shows possible solutions, symptom frequency, symptom details, and which files contained them

Global Score	Type	Symptom	Symptom Occurrences	Knowledge Base Matches	First Occurrence Timestamp	Last Occurrence Timestamp	ID
	ErrorMsg	SRVE0255E A WebGroup/Virtual Host to handle /favicon.ico has not been defined.	2	10	06/08/10 14:02:18:843	06/08/10 14:02:21:843	11
	ErrorMsg	HMGR0028E A duplicate DCS_UNICAST_ADDRESS port has been detected. Members bullisCell02/bullisNode02/nodeagent and bullisCell02/bullisNode02/server2 on host bullis.austin.ibm.com are both configured to use port 9356.	1	10	06/08/10 13:54:57:000	06/08/10 13:54:57:000	3
	ErrorMsg	SECJ0350E Could not get the uniqueId of the user samples.	1	10	06/08/10 13:55:05:515	06/08/10 13:55:05:515	4

Knowledge Base Matches Symptom Occurrences Symptom Details Containing Files

Expand All **Find More Results**

- ▶ APAR: PK85685. THE ERROR MESSAGE SRVE0255E RETURNS TO A CLIENT BROWSER IF THE APPLICATION IS DOWN. [LocalKBSearch]
- ▶ APAR: PM27878. SRVE0255E: A WEBGROUP/VIRTUAL HOST TO HANDLE /IBM/IMAGES/ATTEND.GIF HAS NOT BEEN DEFINED. [LocalKBSearch]
- ▶ APAR: PK77176. WEBCONTAINER FAILS TO MAP REQUEST PROPERLY, IF THE HOST NAME IN ALIAS HAS FEWER THAN 2 CHARACTERS, LEADING TO 404 ERROR. [LocalKBSearch]
- ▶ APAR: PK65519. ADMINISTRATIVE CONSOLE SHOULD SUPPORT ENTRY OF [] BRACKETS AROUND IPV6 ADDRESSES. [LocalKBSearch]

© Copyright IBM Corporation 2013

Figure 2-34. Automated analysis: Symptoms tab

WA5913.0

Notes:

The top portion of the Symptoms tab shows a list of all known symptoms found in the diagnostic files during the scan. It includes a symptom summary, confidence ranking, number of times it was found, and how many potential solutions were found in the KB.

The four tabs at the bottom of the window provide more details. The “Knowledge Base Matches” tab shows APARs and technotes that might be applicable to the symptom. Clicking on an entry shows more information along with the APAR/technote content and links to other information.

The “Symptom Occurrences” tab shows information for each occurrence of the selected symptom including when it occurred and in which server.

The “Symptom Details” tab shows a variety of information about the symptom and information about the automated analysis regarding the symptom.

The “Containing Files” tab shows which files contain the symptom along with other information (For example, runtime environment, product, and others) contained in the files.



Automated analysis: Knowledge tab

- Table of potential solutions (APARs)
- Details of each solution
- Applicable symptom occurrences for each solution
- Files containing the symptoms applicable to the potential solution

Global Score	Type	Knowledge Base Entry	Symptom	Tool	ID
	APAR	PK75700 DCS_UNICAST_ADDRESS PORT CONFLICT RESULTS IN HMGR0028E, DCSV1036W AND EVENTUALLY JAVA OUTOFMEMORYERROR IN THE SERVER.	Multiple symptoms (2) matched by this entry	LocalKBSea	11
	APAR	PK85685 THE ERROR MESSAGE SRVE0255E RETURNS TO A CLIENT BROWSER IF THE APPLICATION IS DOWN	SRVE0255E A WebGroup/Virtual Host to handle /favicon.ico has not been defined.	LocalKBSea	41

Knowledge Base Matches Symptom Occurrences Symptom Details Containing Files

Type: APAR
Found by: Tool: LocalKBSearch
Global Score: 2715
Label: PK75700 DCS_UNICAST_ADDRESS PORT CONFLICT RESULTS IN HMGR0028E, DCSV1036W AND EVENTUALLY JAVA OUTOFMEMORYERROR IN THE SERVER.
Match ID: 11
Symptom IDs associated with this Match: 3,6
Description:
Click on the link for more references:
<http://www.ibm.com/Search/?q=PK75700>
Abstract:
DCS_UNICAST_ADDRESS PORT CONFLICT RESULTS IN HMGR0028E, DCSV1036W AND EVENTUALLY JAVA OUTOFMEMORYERROR IN THE SERVER.
Error description:
When running a configured new server with an already
'in-use' port for the DCS_UNICAST_ADDRESS, the HA manager will
detect the port conflict by issuing HMGR0028E messages.
The HA Manager communication amongst these server will
eventually result in Java OutOfMemoryError for the control
region. When checked the heap had lots of DCS Alarm() objects.
The heap usage looked something like this.
SECTION MEMINFO subcomponent dump routine

© Copyright IBM Corporation 2013

Figure 2-35. Automated analysis: Knowledge tab

WA5913.0

Notes:

When the global filter is on and the “Flat View” of the Files tab is selected, the list of files is limited by the filter criteria.

Files that contain multiple events such as restarts are segmented into those time frames.

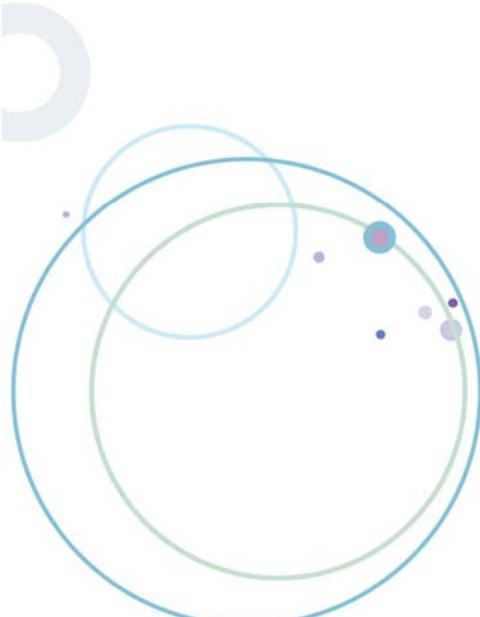
The “Symptom Occurrences” column tells you how many defined symptoms were found and the “Knowledge Base Matches” tells you how many possible solutions (For example, Technotes, APARs) were found for that symptom.

The Type column shows the type of the diagnostic file. The types are determined by peeking inside the file to determine its type, regardless of its file name extension. For example, this lets you tell whether a .log file is from WebSphere or DB2.

When you click a row in the table of files, details about that file are displayed below the table.

2.6. Installing IBM Support Assistant 5

Installing IBM Support Assistant 5



© Copyright IBM Corporation 2013

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

8.0

Figure 2-36. Installing IBM Support Assistant 5

WA5913.0

Notes:

Installing IBM Support Assistant 5: Overview

- Installation option: Use the Installation Manager
 - Choose which problem determination tools to install
 - Makes it easy to keep IBM Support Assistant and tools up-to-date
- Installation option: All-in-one compressed file
 - Includes IBM Support Assistant and all available problem determination tools
 - Ideal for environments that are disconnected from the Internet
- Deployment option: Embedded server (Preferred)
 - Embedded server
 - Configuration ready to run
- Deployment option: EAR deployment to an existing WebSphere Application Server environment
 - Flexibility

© Copyright IBM Corporation 2013

Figure 2-37. Installing IBM Support Assistant 5: Overview

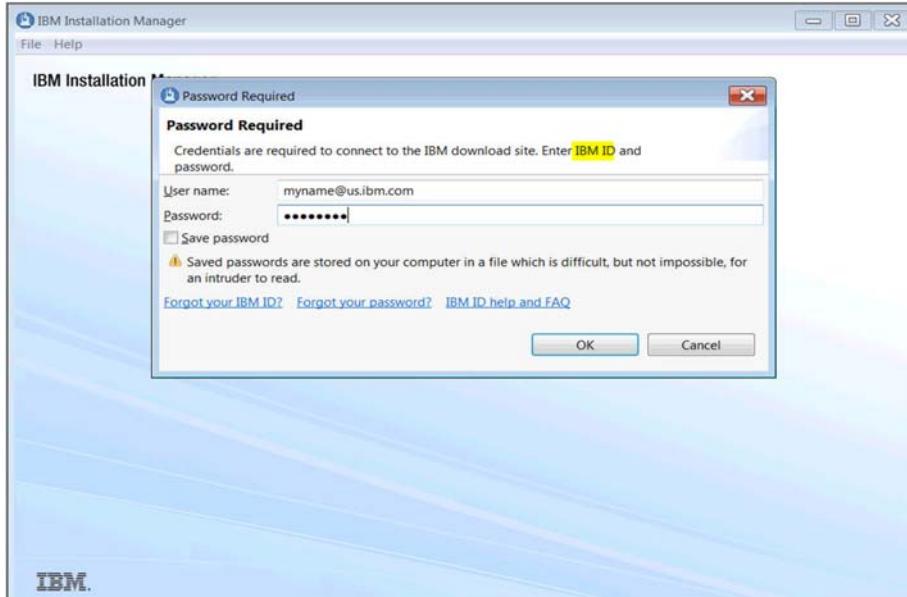
WA5913.0

Notes:



Installation Manager: IBM ID credentials

- Download ISA 5: <http://www.ibm.com/software/support/isa/teamserver.html>
- Extract Installation Manager package for IBM Support Assistant.
- Start Installation Manager.



© Copyright IBM Corporation 2013

Figure 2-38. Installation Manager: IBM ID credentials

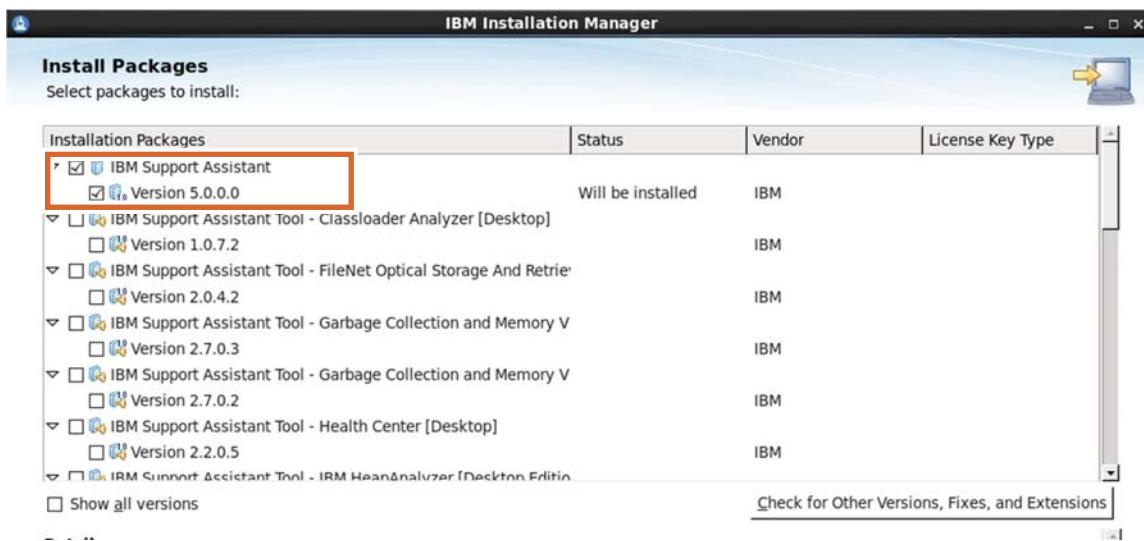
WA5913.0

Notes:

When you download ISA 5 for installing with IM the repository location of IBM Support Assistant is configured for you. When you start IM you are prompted for your IBM ID and password.

Installation Manager: Install packages

- Choose which packages (ISA and PD tools) to install.
- Upgrade Installation Manager if necessary



© Copyright IBM Corporation 2013

Figure 2-39. Installation Manager: Install packages

WA5913.0

Notes:

The first pane you see lets you choose to install ISA along with any optional problem determination tools. ISA is the only selection that is required.

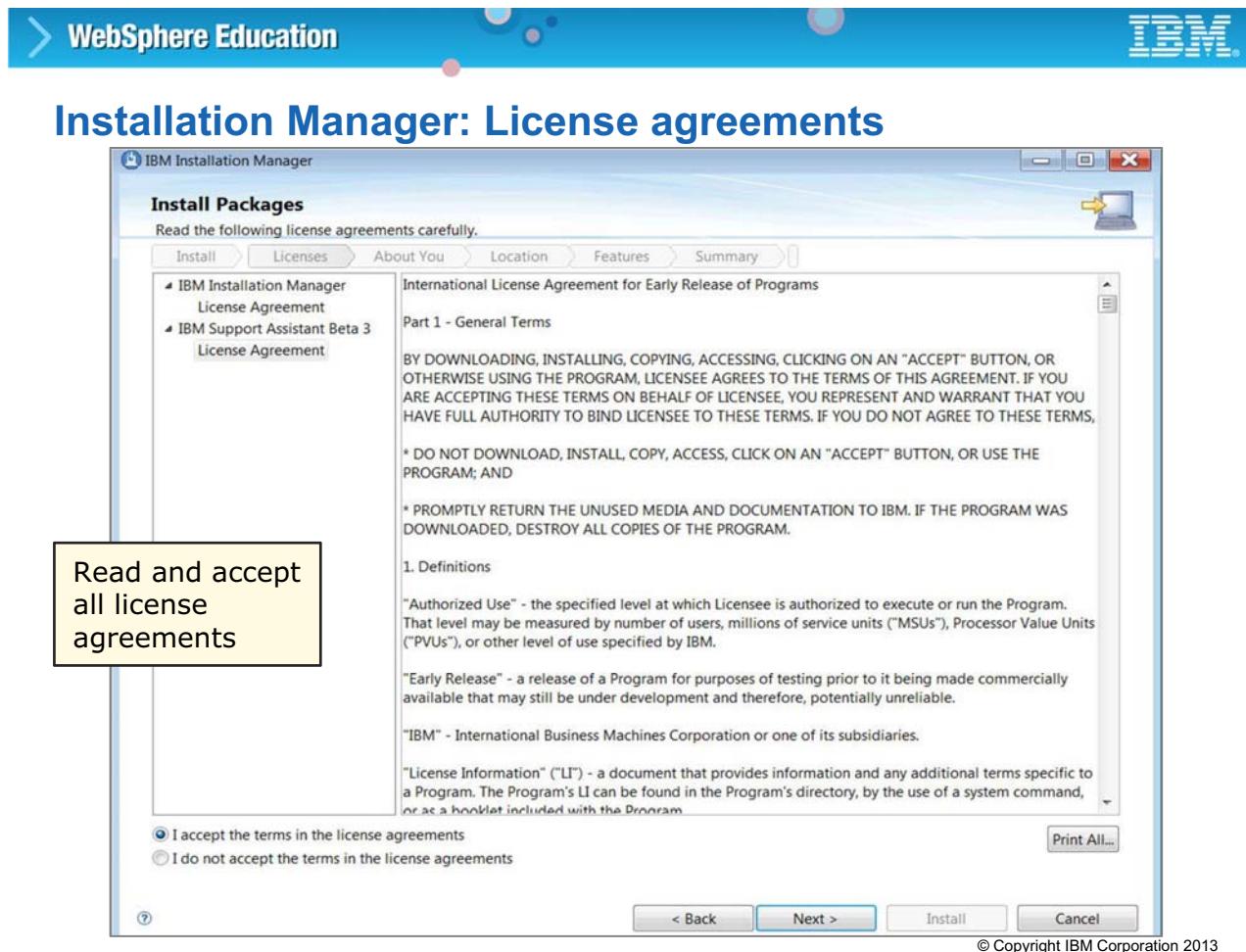


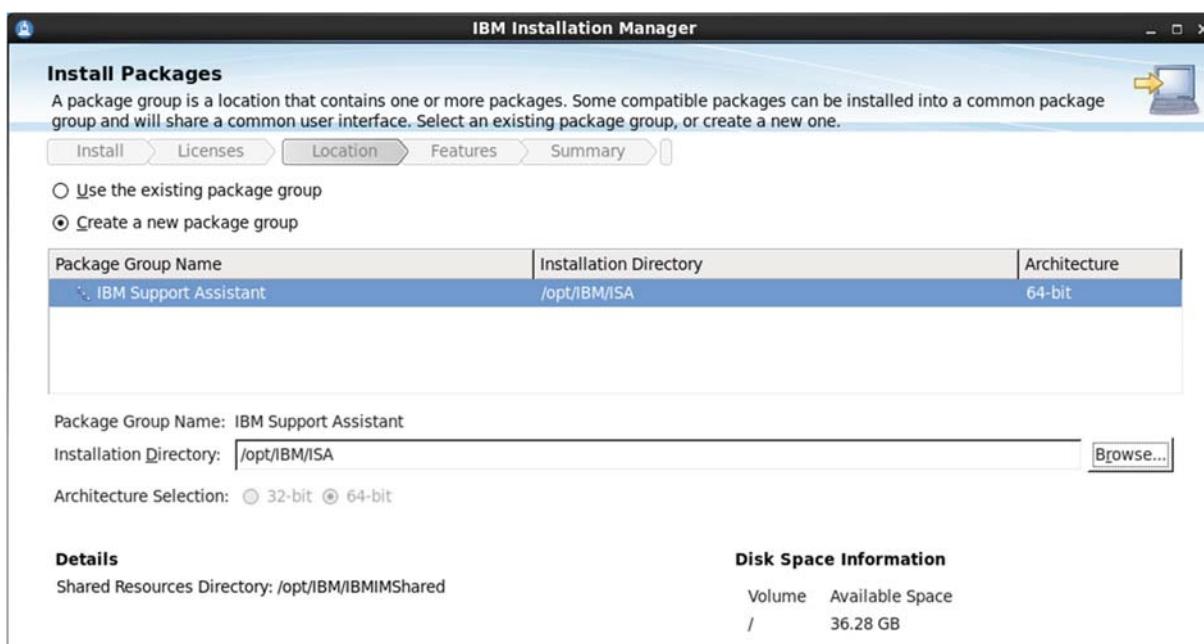
Figure 2-40. Installation Manager: License agreements

WA5913.0

Notes:



Installation Manager: Create package group



© Copyright IBM Corporation 2013

Figure 2-41. Installation Manager: Create package group

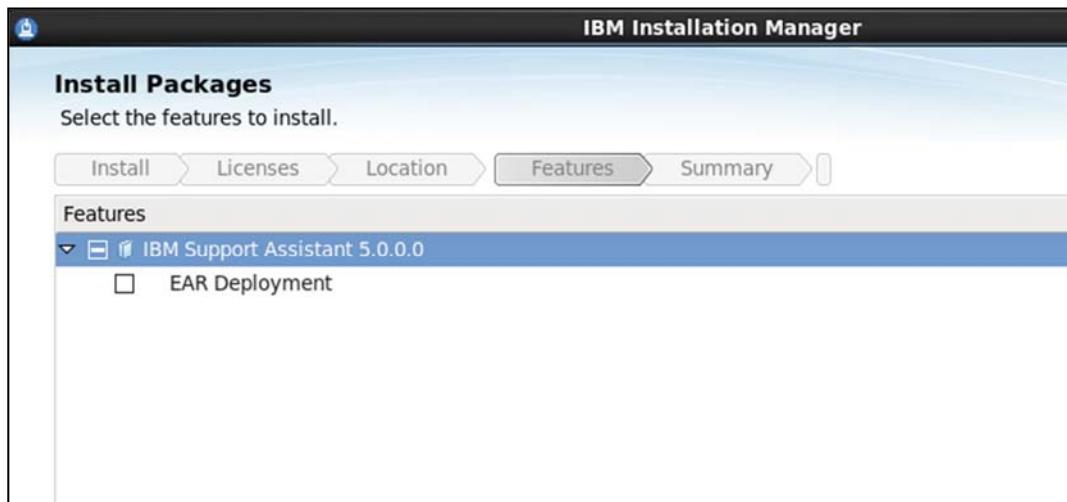
WA5913.0

Notes:



Installation Manager: Select features

- Select IBM Installation Manager, if necessary
- Select IBM Support Assistant



© Copyright IBM Corporation 2013

Figure 2-42. Installation Manager: Select features

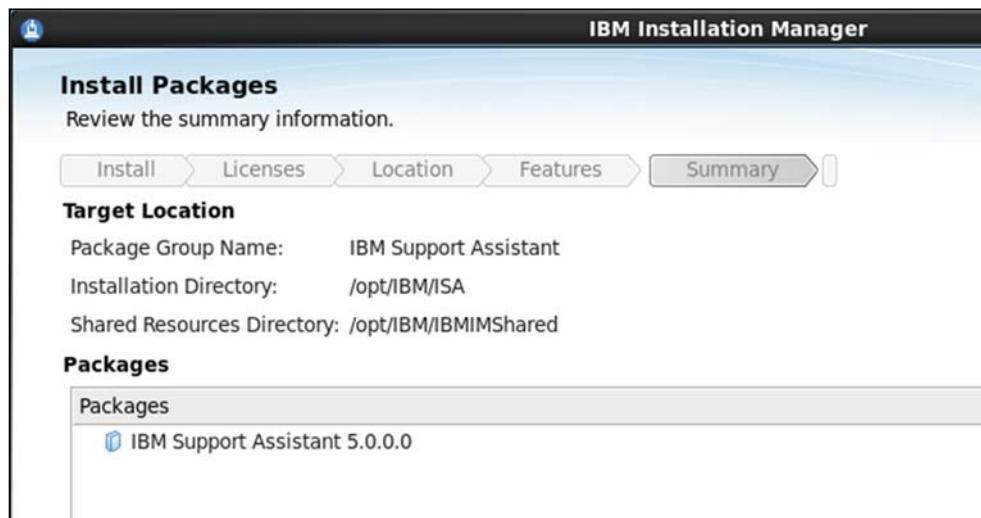
WA5913.0

Notes:



Installation Manager: Confirm installation packages

- Confirm that IBM Installation Manager is selected for installation
 - If necessary
- Confirm that IBM Support Assistant is selected for installation



© Copyright IBM Corporation 2013

Figure 2-43. Installation Manager: Confirm installation packages

WA5913.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe the various components of IBM Support Assistant Team Server
- Install, update, and remove problem determination tools
- Manage diagnostic files with cases
- Launch problem determination tools
- View reports that are generated during problem analysis
- Perform automated analysis and review results
- Install the IBM Support Assistant Team Server 5.0

© Copyright IBM Corporation 2013

Figure 2-44. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. (T/F) The Tool Administration tab allows you to install and update problem determination tools.
2. (T/F) The case management feature is used to generate Java heap dumps from an application server.
3. (T/F) There are three types of problem determination tools: report generators, interactive web tools, and desktop tools.
4. (T/F) The results of an automated analysis are stored in the Reports tab.

© Copyright IBM Corporation 2013

Figure 2-45. Checkpoint questions

WA5913.0

Notes:



Checkpoint answers

1. True
2. False: The case management feature provides a container for diagnostic files.
3. True
4. False: The results of an automated analysis are stored in the Overview, Symptoms, and Knowledge tabs.

© Copyright IBM Corporation 2013

Figure 2-46. Checkpoint answers

WA5913.0

Notes:

Exercise 1

Using the IBM Support Assistant Team Server 5.0

© Copyright IBM Corporation 2013

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

8.0

Figure 2-47. Exercise 1

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Start and stop the IBM Support Assistant Team Server
- Administer the IBM Support Assistant Team Server
- Run report generator tools and examine the reports
- Use the Case Manager to create a case and add diagnostic data
- Run interactive desktop tools
- Use Automated Analysis to scan a case and examine the results
- Run the IBM Support Assistant Data Collector tool

© Copyright IBM Corporation 2013

Figure 2-48. Exercise objectives

WA5913.0

Notes:

Unit 3. Problem determination methods

What this unit is about

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

What you should be able to do

After completing this unit, you should be able to:

- Prepare for problems before they occur
- Characterize a problem from its symptoms
- Collect diagnostic data
- Implement a relief or recovery plan
- Troubleshoot a problem

How you will check your progress

- Checkpoint questions

References

IBM Redbooks: www.redbooks.ibm.com

Redbooks publication SG24-6798-00: WebSphere Application Server V6
Problem Determination for Distributed Platforms:
[http://www.redbooks.ibm.com/abstracts/
sg246798.html](http://www.redbooks.ibm.com/abstracts/sg246798.html)
is updated for V6.1

Redpaper: WebSphere Application Server V6.1: Workload Management
Problem Determination
[http://www.redbooks.ibm.com/abstracts/
redp4308.html](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](http://www.redbooks.ibm.com/redpieces/abstracts/REDP4330.html)

12 ways you can prepare for effective production troubleshooting:
[http://www.ibm.com/developerworks/websphere/
techjournal/0708_supauth/0708_supauth.html](http://www.ibm.com/developerworks/websphere/techjournal/0708_supauth/0708_supauth.html)

Unit objectives

After completing this unit, you should be able to:

- Prepare for problems before they occur
- Characterize a problem from its symptoms
- Collect diagnostic data
- Implement a relief or recovery plan
- Troubleshoot a problem

© Copyright IBM Corporation 2013

Figure 3-1. Unit objectives

WA5913.0

Notes:

Topics

- Preparation: Before problems occur
- Organize the investigation
- Relief options
- Phase 1: Initial investigation
- Phase 2: In-depth investigation

© Copyright IBM Corporation 2013

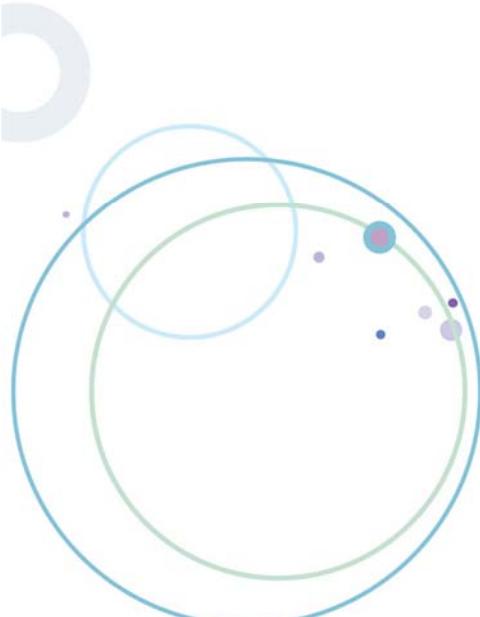
Figure 3-2. Topics

WA5913.0

Notes:

3.1. Preparation: Before problems occur

Preparation: Before problems occur



© Copyright IBM Corporation 2013

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

8.0

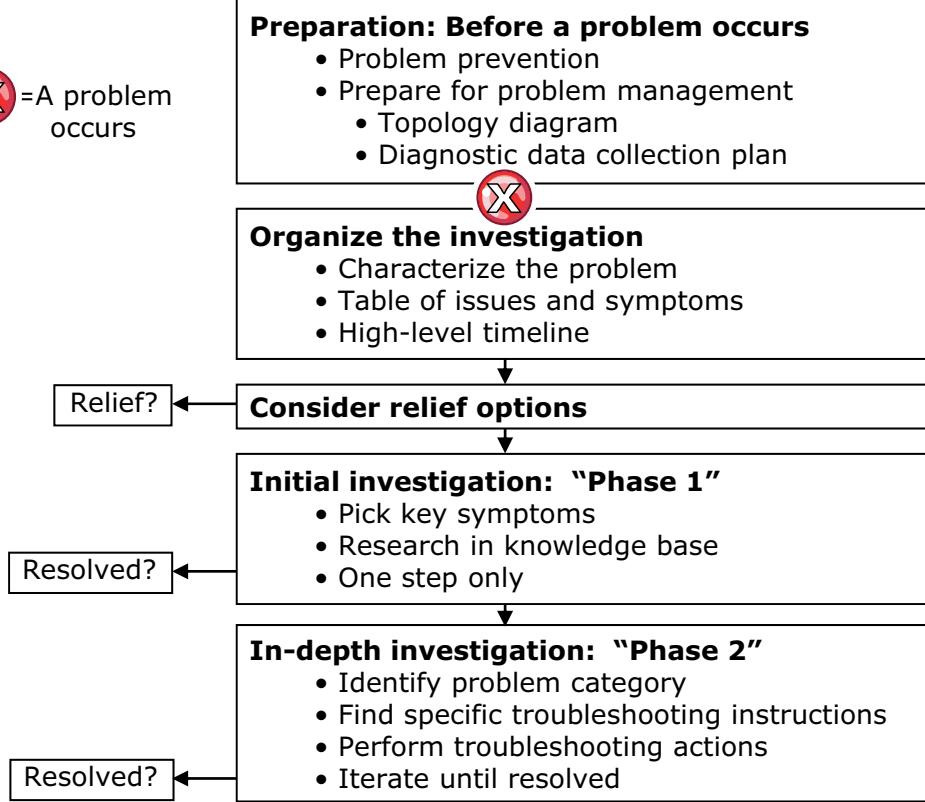
Figure 3-3. Preparation: Before problems occur

WA5913.0

Notes:

Key steps for problem determination

 = A problem occurs



© Copyright IBM Corporation 2013

Figure 3-4. Key steps for problem determination

WA5913.0

Notes:

The red X represents the point at which a problem occurs. The job of a troubleshooter does not begin after a problem occurs. Plenty of preparation work 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, the system fails), and they dive into 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 explained later in this presentation.

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

- Quickly provide a temporary solution so that users can get back to work and not have the problem continue to affect them, while you spend more time in looking for the permanent solution.
- Find and implement the right, permanent solution.

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

Prepare for effective production troubleshooting

- 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

© Copyright IBM Corporation 2013

Figure 3-5. Prepare for effective production troubleshooting

WA5913.0

Notes:

See the developerWorks article: “The Support Authority: 12 ways you can prepare for effective production troubleshooting”:

http://www.ibm.com/developerworks/websphere/techjournal/0708_supauth/0708_supauth.html

Keep a change log.

An important aspect of most troubleshooting exercises is to figure out what is different between a working system and a broken one, and using a baseline is one way to help address the question. Another action that can help you determine system differences is to keep a rigorous log of all changes that are applied to the system over time. When a problem occurs, you can look back through the log for any recent changes that possibly contributed to the problem. You can also map these changes to the various baselines that were collected in the past to ascertain how to interpret differences in these baselines.

Your log should at least track all upgrades and software fixes that are 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 or a different mix of operations that users invoke.

In a complex IT environment where many teams contribute to different parts of the environment, the task of maintaining an accurate, up-to-date, and global change log can be surprisingly difficult. You can use tools and techniques to help this task, from collecting regular snapshots of the configuration with simple data collection scripts, which are used to collect diagnostic data, to sophisticated system management utilities.

It is important to know the concept of change control, and keeping a change log is generally broader than the troubleshooting arena. Change control is also considered one of the key best practices for managing complex systems to prevent problems, as opposed to troubleshooting them.

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, for example)
- 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 get detected only indirectly through some secondary effect. You need tools and a plan to effectively detect problems or any potential anomalies when they emerge.

System documentation: When investigating a problem, you need much 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 must quickly and effectively gather information that allows you to diagnose the problem. To do so, you need a plan that is set up *in advance* for what diagnostics you 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 users. The relief or recovery plan lays out, in advance, the steps that you undertake to restore this function, without knowing in advance exactly what problem occurs.

Maintenance: Applying regular maintenance is one of the key factors to reduce the probability and impact of problems. The maintenance plan establishes how you do so regularly. In addition to

regular scheduled maintenance, you also must make emergency changes or maintenance to the system, in response to a newly diagnosed problem. The emergency maintenance plan outlines how to do so safely and effectively.

Change log: Your log should at least track all upgrades and software fixes that are applied in every software component in the system, including both infrastructure products and application code. It should 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, or a different mix of operations that users invoke.

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

Monitoring and problem detection

- Often, problems go undetected for a long time, or get detected only indirectly through some secondary effect
- Monitoring tools and a plan are needed to effectively detect problems or anomalies when they emerge
 - Trade-off: Detect important events, but do not degrade performance
- Passive monitoring versus active monitoring
 - **Passive:** Examine logs, PMI statistics, heap size, verbose GC data
 - **Active:** Send a test transaction from end to end
- Watch low-level operating system and network metrics
 - Overall processor and memory usage for the entire server
 - Paging and disk I/O activity
 - Rate of network traffic between various components
- Be prepared to actively generate more diagnostic data when a problem occurs

© Copyright IBM Corporation 2013

Figure 3-6. Monitoring and problem detection

WA5913.0

Notes:

Monitoring tools and a plan are needed to effectively detect problems or anomalies 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.

Passive monitoring can be done at all levels: network, operating system, application server, application. Dependent systems such as databases and LDAP directories can also be monitored. Monitor the main system log files for errors and events. For example, you might detect application server restarts that indicate that the server is failing. Some tools for passive monitoring include the Tivoli Performance Viewer and IBM Tivoli Composite Application Manager for Application Diagnostics.

Active monitoring goes beyond passive monitoring. Periodically test the operation of the entire system from end to end. One technique that can be used is the pinging of system components, such as one server or one database connection. Another technique is end-to-end ping: periodically sending an entire “dummy” transaction through the system and verifying that it completes. Some tools for active monitoring include IBM Tivoli Composite Application Manager for Transactions and web-based load-generating programs like Rational Performance Tester.

Be prepared to actively generate more diagnostic tests when a problem occurs. In addition to dealing with diagnostic artifacts that are present when an incident occurs, your troubleshooting plan should consider any additional explicit actions to take as soon as an incident is detected. You want these actions to take place before the data disappears or the system is restarted.

Here are some examples of explicit actions to generate more diagnostic messages:

- Actively trigger various system dumps, if they are generated automatically (such as Java dump, heap dump, system dump, WebSphere Application Server Diagnostic Provider dumps, or other memory dumps that various products and applications might provide). 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, and processor usage.
- Enable and collect information from the WebSphere Application Server Performance Monitoring Infrastructure instrumentation.
- Dynamically enable a specific trace, and collect that trace for a specified interval while the system is in the current unhealthy state.

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
- Be specific and detailed
 - Indicate software versions, server names, and IP addresses, if possible
- 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

© Copyright IBM Corporation 2013

Figure 3-7. System architecture or topology diagram

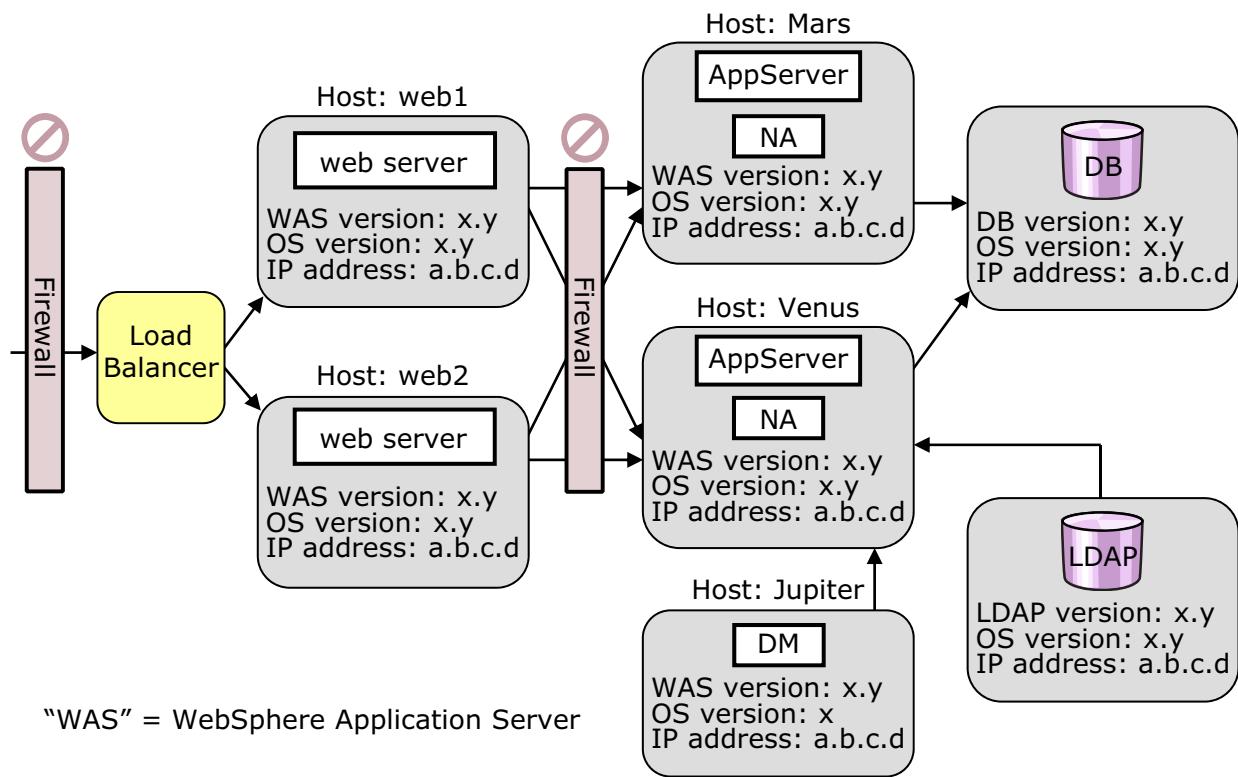
WA5913.0

Notes:

One suggestion is to meet with all the parties involved in the deployment and operation of the system together to draw or redraw the system topology diagram.

- Often, the mere fact of having multiple groups compare their understanding of the topology of the system yields important clues and dependencies that the various groups maybe did not know about 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.

Example: Topology diagram



© Copyright IBM Corporation 2013

Figure 3-8. Example: Topology diagram

WA5913.0

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
- Be specific:
 - Machine names and IP addresses.
 - Software versions of all software that is installed on each machine.
 - Do not forget the network topology and relationships (a separate network topology chart might be useful).

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

Establish baselines

- What does a “normal” system look like?
- What are the typical values for common system metrics?
 - Processor usage
 - Memory size
 - PMI statistics
- What is a typical load?
- What is a typical response time for various operations?
- What should you normally see in the logs during operation of the system?
- Understand and document the system baseline

© Copyright IBM Corporation 2013

Figure 3-9. Establish baselines

WA5913.0

Notes:

Much of what you do for problem determination is to observe various aspects of the behavior of the system. Then, try to determine whether these behaviors are normal or if they represent a symptom of a possible problem.

In many actual systems, it is not unusual to see various 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 and trace files, over a representative time period 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 system that is working correctly, before a problem occurs.
- Information about the normal transaction rates in the system and response times.

- Various operating system level statistics on a correctly running system, such as processor usage, memory usage, and network traffic.
- Copies of any other artifacts, information, or normal expected results from any of the special diagnostic collection actions, suggested earlier, for each anticipated type of problem.

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
 - Gathering data 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 (SystemOut, SystemErr, native_stdout, native_stderr)
 - WebSphere first-failure data capture (FFDC) facility
 - HTTP access logs (what are default log detail levels?)
 - Core memory dumps, javacore memory dumps, heap memory dumps
 - Enable verbose garbage collection
 - Minimal monitoring of operating system resources
 - Minimal monitoring of network health
 - Application logging

© Copyright IBM Corporation 2013

Figure 3-10. Diagnostic data collection plan (1 of 2)

WA5913.0

Notes:

Diagnostic tools to consider:

- JVM verboseGC log: often useful, and usually relatively low processor usage on a well-tuned system.
- JVM Java dumps, heap dumps, and system dumps: Java dumps are typically cheap to produce, and can be enabled for automatic generation. Heap dumps and system dumps can involve significant processor usage, so consider carefully before setting them up to be triggered automatically.
- Increased logging of requests 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 that the WebSphere Application Server Performance Monitoring Infrastructure provides.
- 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 minimal impact on performance. The benefit usually outweighs the cost.

Each of these items is explained later in the presentation, or covered in detail later in the course.

Diagnostic data collection plan (2 of 2)

- Identify active diagnostic actions to take for the most likely problems
- For example:
 - Trigger a javacore or system core memory dump
 - Trigger a heap memory dump
 - Attempt pings and application checks
 - Collect specific PMI statistics
 - Use Java Health Center to monitor an active JVM
- Make arrangements so that all the predetermined diagnostic elements get reliably captured
 - Clearly identify where they are
 - Established procedures to collect them and practice
 - Synchronize the clocks between all servers to facilitate analysis
 - Manage the logs and all other diagnostics artifacts to avoid surprises during normal operation
 - Make sure that you have enough disk space available for diagnostic artifacts

© Copyright IBM Corporation 2013

Figure 3-11. Diagnostic data collection plan (2 of 2)

WA5913.0

Notes:

These “active” diagnostic collection actions resemble the ones that were already mentioned when explaining regular monitoring (as in pinging). However, they are actions that you 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.

Java Health Center is a low monitoring tool that does not involve a prohibitive amount of processor usage. It runs alongside an IBM Java application with a small impact on the application’s performance. For more information, see:

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

Do not forget the human element:

- Who is in charge, when there is a production problem, of implementing the diagnostic data collection plan and taking recovery actions?
- What groups must be informed or coordinated with?

The simplest diagnostic collection plan is in the form of plain, written documentation that lists all the detailed manual steps that must be taken. To be more effective, try to automate as much of this plan as possible. Provide one or more command scripts that can be invoked to perform a complex set of

actions, or use more sophisticated system management tools. The various collector tools and scripts now offered as part of IBM Support Assistant can provide a good framework for you to start automating many diagnostic collection tasks.

Maintenance plans

- Applying regular maintenance (interim fixes, fix packs) reduces the probability and impact of problems
 - The **maintenance plan** describes how to apply fixes regularly
- In addition to regular scheduled maintenance, you sometimes must apply emergency maintenance to the system, in response to a newly diagnosed problem
 - The **emergency maintenance plan** outlines how to do this task safely and effectively
- Maintenance is at all levels:
 - Application server and each of the products that are involved in the system
- Keep records of current maintenance levels on the topology diagram
 - Have processes for verifying the maintenance levels regularly
- Consider upgrading to the latest available fix pack during an investigation

© Copyright IBM Corporation 2013

Figure 3-12. Maintenance plans

WA5913.0

Notes:

Consider upgrading to the latest available fix pack during an investigation. Individual fixes are meant to be temporary until a fix pack is available. There is considerable risk in using too many individual fixes because it is not possible to test all the possible interactions among individual fixes.

Because of the complexity of the system and difficulty of reproducing problems and gathering diagnostic information, it is not always practical to determine exactly which fix (APAR) resolved a particular situation.

Overall, it is suggested to have a strong maintenance strategy.

Use Fix Central to download fixes:

<http://www.ibm.com/support/fixcentral>

3.2. Organize the investigation

Organize the investigation



© Copyright IBM Corporation 2013

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

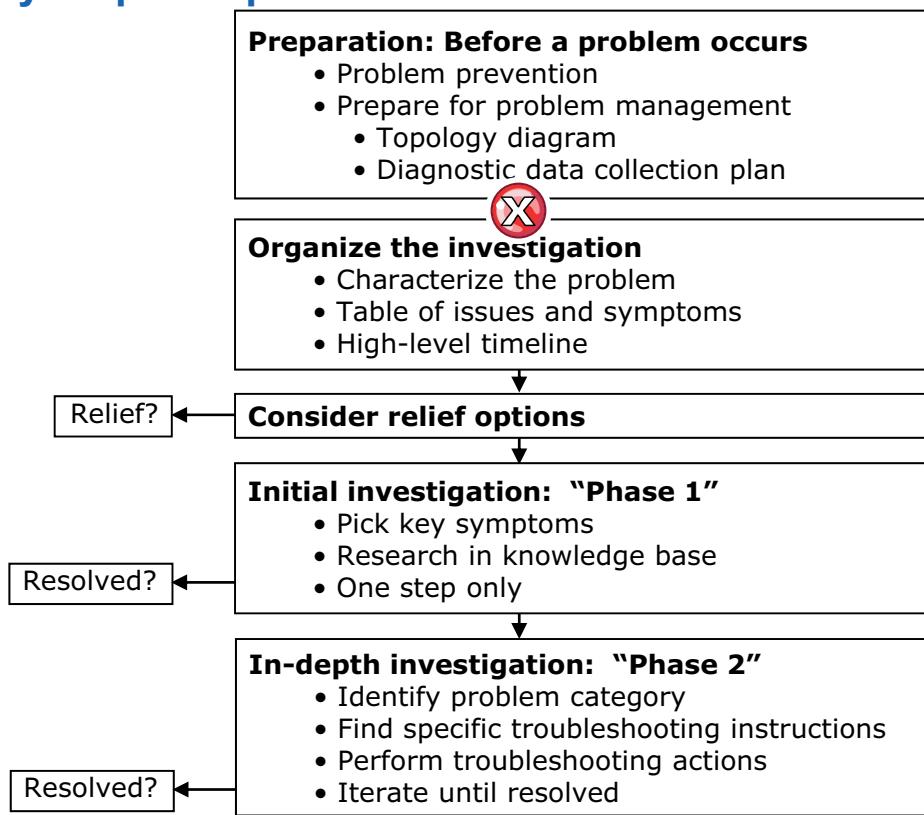
8.0

Figure 3-13. Organize the investigation

WA5913.0

Notes:

Key steps for problem determination



© Copyright IBM Corporation 2013

Figure 3-14. Key steps for problem determination

WA5913.0

Notes:

Characterize the problem (1 of 2)

- Take the time to carefully understand the problem and its context
- Listen and ask questions
 - In many cases, asking questions is all that it takes to solve simple problems
 - For complex problems, failure to do so results in considerable delays
- Ask: **What? Where? When? and Why?**
- Develop a clear and specific description of **what** happened, error messages, observed abnormal behavior, and other symptoms
 - How would you recognize the same problem if it happens again?
 - What exactly would be different after 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*
 - Be alert for possibly unrelated symptoms, and the possibility that several independent problems occur at the same time
- **Where** exactly did the problem occur: What system? What server?

© Copyright IBM Corporation 2013

Figure 3-15. Characterize the problem (1 of 2)

WA5913.0

Notes:

There is a good summary in G. Polya, "How to Solve It", Second edition, Princeton University Press, 1957, ISBN 0-691-08097-6, at the following website:

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

It explains a basic methodology for problem solving.

Characterize the problem (2 of 2)

When exactly did the problem occur?

- Did it happen only one time, or does it happen repeatedly?
- If the problem happens repeatedly, characterize the circumstances
 - Apparently random times?
 - What frequency?
 - Every time that you do X, or every time some other external event happens?

Consider why this problem occurred here and now, and not in the past

- Are you attempting something new for the first time?
 - First product installation?
- Has anything changed in the environment?
 - Configuration changes in the failing system
 - Changes to dependent systems
- Does this problem occur in similar environments?

© Copyright IBM Corporation 2013

Figure 3-16. Characterize the problem (2 of 2)

WA5913.0

Notes:

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

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

Example: Table of issues and symptoms

No	Status, priority	Symptom or issue	Plan of action, disposition
#1	Open/High	AppServer unresponsive to HTTP requests	Awaiting result of investigation on #8
#9	Open/Medium	Large number of HTTP processes: observed during #1	Defer
#8	Open/High	"CONM6026W: Timed out waiting for connection from data source": observed during #1	Enable connection pool diagnostic tracing
#5	Open/Medium	Application error: "Cannot validate credit card"	Awaiting feedback from application developer
#2	Open/High	AppServer crashes: happens sometimes after #1, but not always	Awaiting result of investigation on #6
#6	Open/High	verboseGC reports allocation failure for large (2BM) object: last entry in log before #2	Capture heap memory dump

© Copyright IBM Corporation 2013

Figure 3-17. Example: Table of issues and symptoms

WA5913.0

Notes:

Developing a table of issues and symptoms is not necessarily a required step, but it is a good idea.

List all top-level issues reported by users of the system, and all low-level symptoms observed during the investigation.

These tables are useful for:

- Organizing the investigation: finding all symptoms and always identifying what to check on next
- Tracking progress
- Making sure that nothing is overlooked

Prioritize and cluster entries to reflect the current state of the investigation while constantly updating and revising as the investigation progresses.

This example is just one way of developing a table of issues and symptoms. This particular example is a complex situation, where suddenly there are many seemingly independent symptoms and several unrelated problems. You do not necessarily use this format. You can customize it to your specific needs.

High-level timeline

List all major events when they occur

- Incidents (occurrences of the same or other problem)
- Data collection actions and experiments
- Relief and remedy actions
- Other maintenance

Useful for:

- Helping to identify patterns, cause, and effect
- Avoiding confusion about what really happened
- Facilitating communication between all parties (customer, IBM Support, and others)
- Record all available diagnostics artifacts and what they correspond to, along with precise time stamps to look for in logs
- Reinforcing change control policies
 - If too many things change at the same time, it might never be possible to understand the problem

© Copyright IBM Corporation 2013

Figure 3-18. High-level timeline

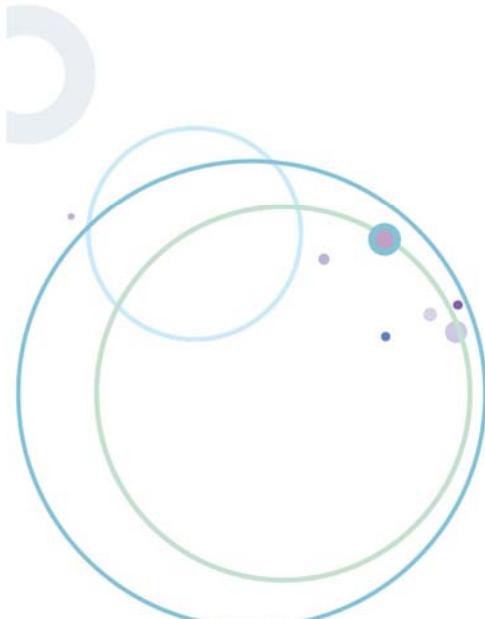
WA5913.0

Notes:

Another suggested practice is to build a high-level timeline. List all major events when they occur. Include incidents, data collection actions and experiments, relief and remedy actions, and other maintenance actions. This practice helps to identify patterns, cause and effect, avoiding confusion about what happened, facilitating communication between all parties, tracking all available diagnostic artifacts and what they correspond to, and reinforcing change control policies.

3.3. Relief options

Relief options



© Copyright IBM Corporation 2013

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

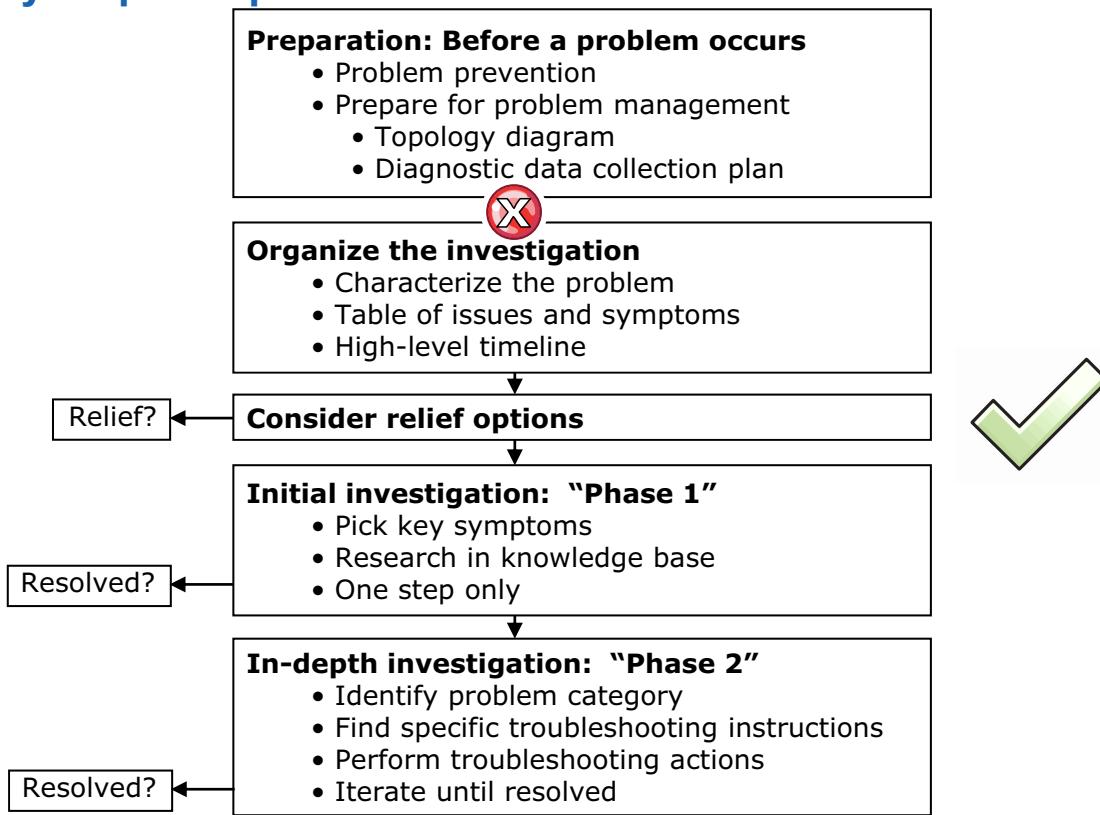
8.0

Figure 3-19. Relief options

WA5913.0

Notes:

Key steps for problem determination



© Copyright IBM Corporation 2013

Figure 3-20. Key steps for problem determination

WA5913.0

Notes:

Relief or recovery plan

- When a problem occurs, a main priority, independent of any investigation, should be to restore functions to the users
- The relief or recovery plan lays out:
 - General steps that you undertake to restore functions
 - Actions to take for specific problems
- Try to predict the most common types of problems, which are based on knowledge of the system topology and flows
 - Loss of an application server
 - Loss of database connectivity
 - Loss of the LDAP server
- Identify different regions of the system that can be isolated or restarted independently
 - Application (Java EE, BLA, OSGi)
 - Application server (JVM)
 - Application server cluster
 - Server

© Copyright IBM Corporation 2013

Figure 3-21. Relief or recovery plan

WA5913.0

Notes:

It is impossible to predict all possible situations in advance; however, much confusion, wasted time, and even greater disasters occur when relief actions are attempted after a problem occurs.
Preparation pays off.

Document the processes:

- Who decides what to do and who does it?
- Have criteria for making such decisions.

Practice the most common relief actions in advance.

- Ensure that you know how to perform them.
- Ensure that they have no unexpected side effects.

Relief options

- If the problem occurs in production or other critical-availability system, you must provide relief long before you fully resolve the problem
- Consider relief timeline:
 - How long can you investigate, before you must provide some relief?
 - Can you keep the system in its failed state, in case you must gather other data later?
- Identify the relief actions:
 - Often you can restart one or more components, but beware of chain reaction effects of stopping and starting some components in a live system
 - Sometimes you can change the usage characteristics of the application by reducing the load and avoiding some “dangerous” operations

© Copyright IBM Corporation 2013

Figure 3-22. Relief options

WA5913.0

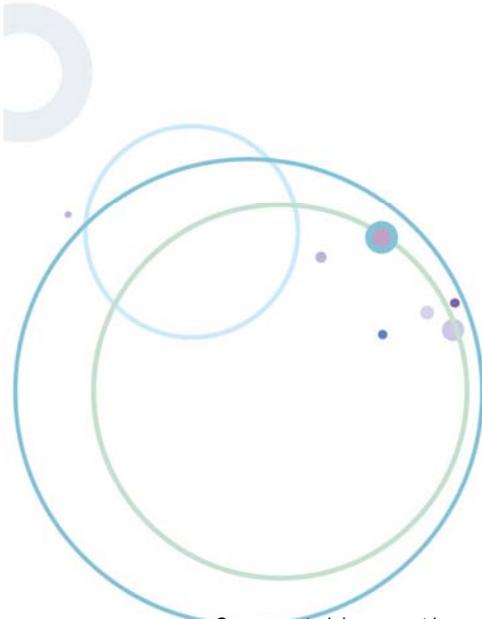
Notes:

You might want to trigger some memory dumps before restarting the system: javacore, system core, or heap.

A “dangerous” operation refers to any function in the application that is deemed likely to trigger a failure, maybe because it is more complex than others, or because it depends on more system components.

3.4. Phase 1: Initial investigation

Phase 1: Initial investigation



© Copyright IBM Corporation 2013

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

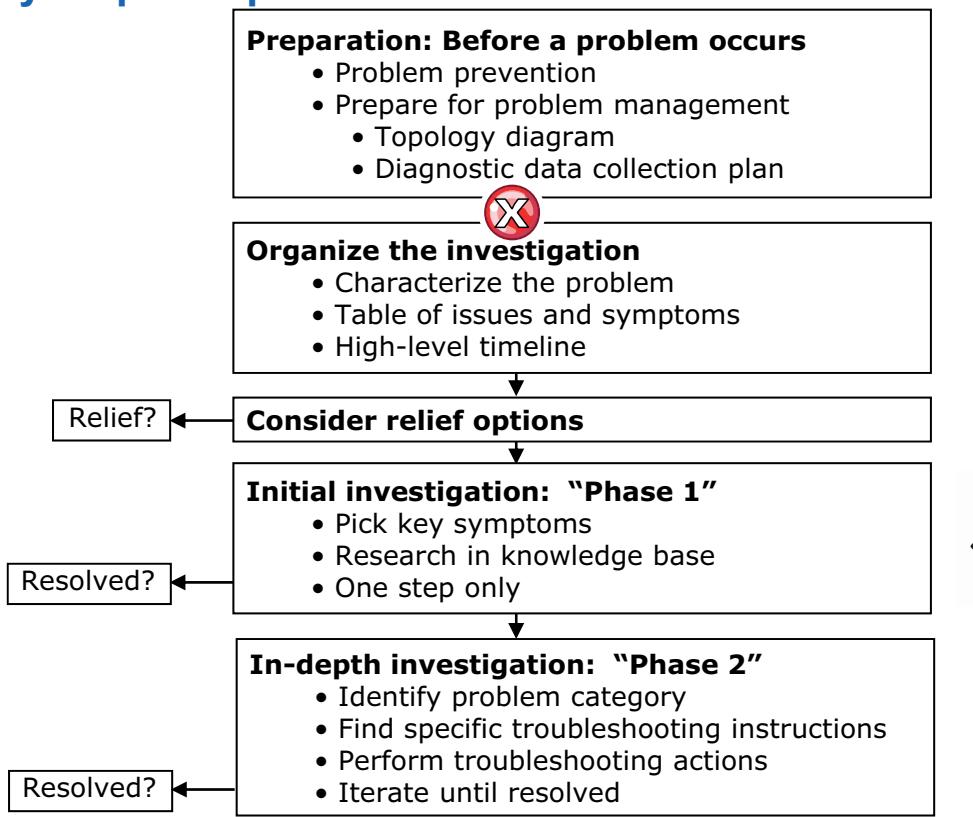
8.0

Figure 3-23. Phase 1: Initial investigation

WA5913.0

Notes:

Key steps for problem determination



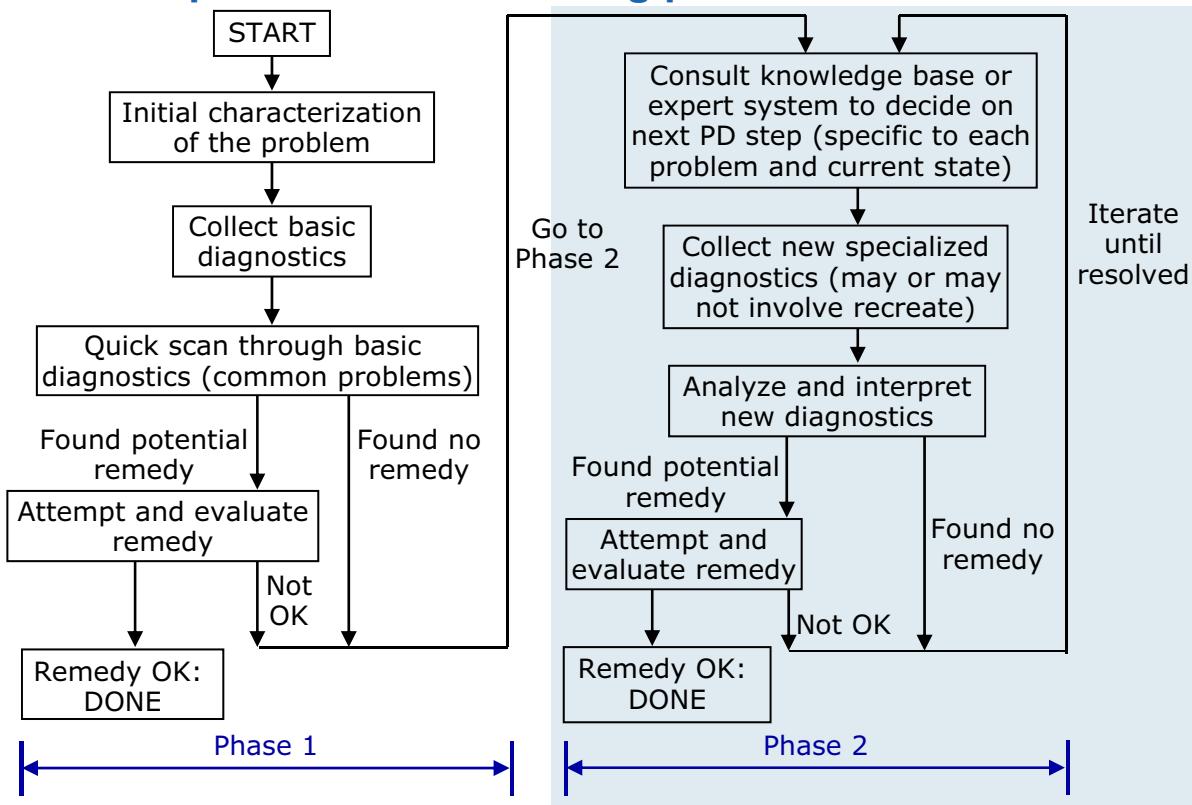
© Copyright IBM Corporation 2013

Figure 3-24. Key steps for problem determination

WA5913.0

Notes:

"Solve a problem" flow: The big picture



© Copyright IBM Corporation 2013

Figure 3-25. "Solve a problem" flow: The big picture

WA5913.0

Notes:

This example is not an “official” process, but for 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. Here the process of resolution differs significantly for each problem.

Phase 1: Initial investigation (1 of 2)

The goal at this stage is to:

- Locate known problems and solutions
- Get a starting point for further 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
- Research that is 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

© Copyright IBM Corporation 2013

Figure 3-26. Phase 1: Initial investigation (1 of 2)

WA5913.0

Notes:

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

Phase 1: Initial investigation (2 of 2)

- Build a low-level, detailed timeline of events for one incident to clarify what might be the cause of what
 - Include pertinent but normal events, in addition to anomalies and symptoms
 - Monitor multiple components in parallel and attempt to correlate
- Research the top symptoms in the WebSphere Knowledge Base
 - Search for relevant IBM technotes
 - Search for APARs
- Answer and verify a favorite question of all troubleshooters: **What changed recently?**
- Clearly communicate between the various parties that are involved in the troubleshooting task
 - Inside your organization
 - When trying to explain a complex environment to IBM support

© Copyright IBM Corporation 2013

Figure 3-27. Phase 1: Initial investigation (2 of 2)

WA5913.0

Notes:

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 suggested solution.

Authorized program analysis report (APAR) tracks software defects reported by customers.

3.5. Phase 2: In-depth investigation

Phase 2: In-depth investigation



© Copyright IBM Corporation 2013

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

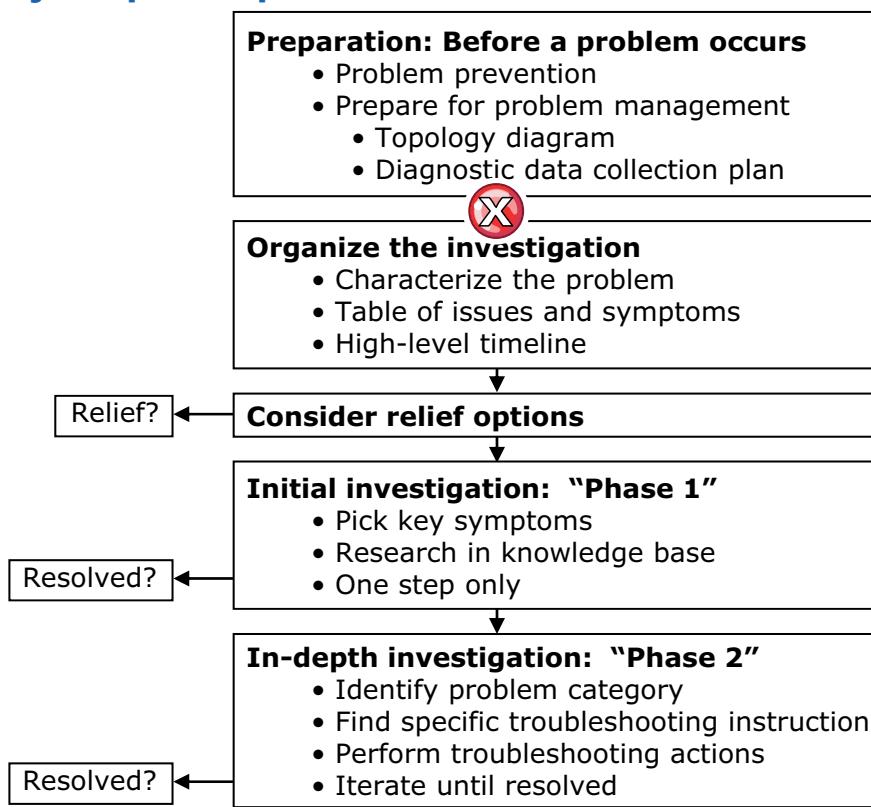
8.0

Figure 3-28. Phase 2: In-depth investigation

WA5913.0

Notes:

Key steps for problem determination



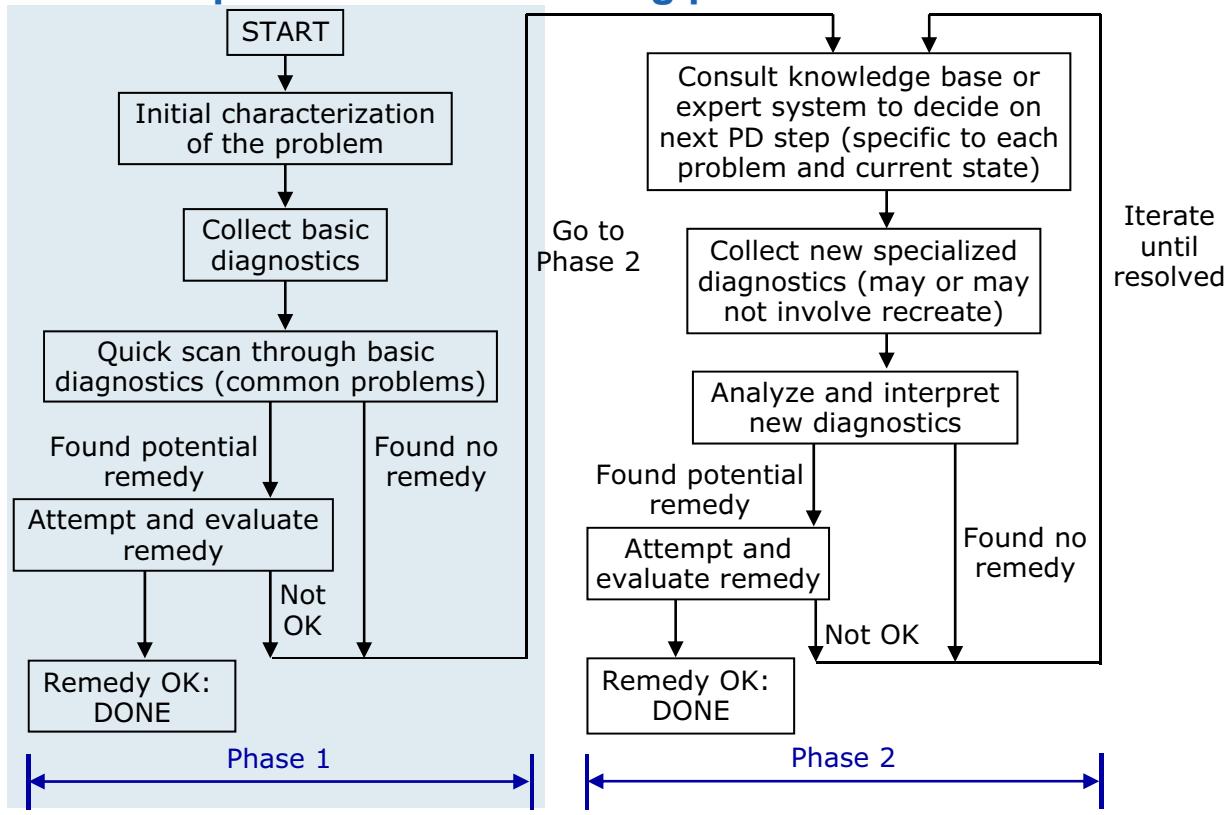
© Copyright IBM Corporation 2013

Figure 3-29. Key steps for problem determination

WA5913.0

Notes:

"Solve a problem" flow: The big picture



© Copyright IBM Corporation 2013

Figure 3-30. "Solve a problem" flow: The big picture

WA5913.0

Notes:

This example is not an “official” process, but for 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. Here the process of resolution differs significantly for each problem.

Phase 2: In-depth investigation

The initial research did not find a solution, so you must undertake a more in-depth investigation

- Using a specific set of methods and tools for each problem

There are several sources of information to start with:

- Troubleshooting section in the information center
- IBM Problem Determination Redbooks
- MustGather documents on the WebSphere Application Server Support website:
 - Define initial set of diagnostic data to focus on (either for IBM support or for internal investigation)
 - From there, search for other technotes with troubleshooting instructions, tools, and other resources
 - The troubleshooting guide on the support website also provides a good starting point for finding relevant documents

© Copyright IBM Corporation 2013

Figure 3-31. Phase 2: In-depth investigation

WA5913.0

Notes:

In this phase, you are looking for specific instructions on how to solve a specific problem.

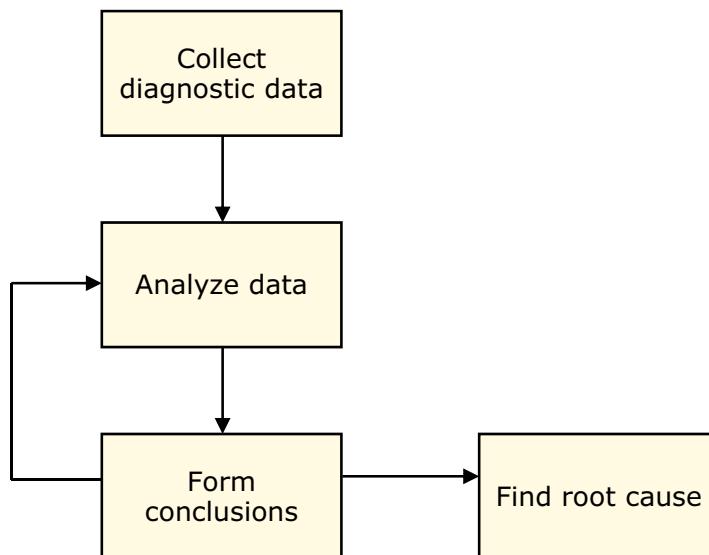
Techniques for gathering information are covered in the next section.

For more information, see:

- IBM Guided Activity Assistant:
<http://www.ibm.com/support/docview.wss?uid=swg24011818>
- Problem determination IBM Redbooks publication:
<http://www.redbooks.ibm.com/abstracts/sg246798.html>

Problem determination techniques (1 of 2)

- Analysis approach
 - Collect and analyze diagnostic data (possibly through several iterations) until the root cause is found



© Copyright IBM Corporation 2013

Figure 3-32. Problem determination techniques (1 of 2)

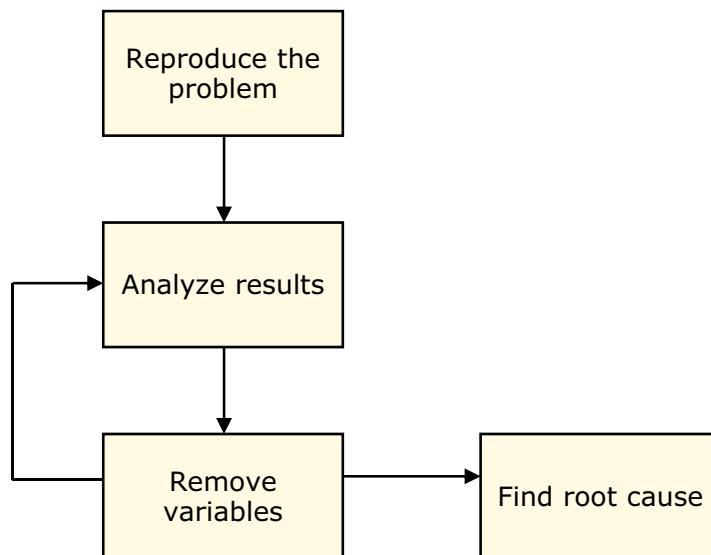
WA5913.0

Notes:

The analysis approach requires you to begin by collecting diagnostic data. Error messages in a log file, for example, might be gathered and analyzed. The messages might indicate that there was an authentication failure. You might recognize immediately from the diagnostic data that the user ID that tried to log in is no longer valid. Or it might be necessary to further analyze the data and eventually discover that the security repository is unavailable.

Problem determination techniques (2 of 2)

- Isolation approach
 - Reproduce the problem, analyze results, and remove variables (possibly through several iterations) until the root cause is found



© Copyright IBM Corporation 2013

Figure 3-33. Problem determination techniques (2 of 2)

WA5913.0

Notes:

A problem with an application can show an error message in a browser; for example, File Not Found. So you prepare to reproduce the problem with tracing enabled on the HTTP plug-in. After you access the application and see the File Not Found message again, you analyze the plug-in trace data. The plug-in trace data shows that the request is being routed to the correct application server. Therefore, you just eliminated the plug-in as a possible problem. You might then make sure that the server is running and that the application is started.

Unit summary

Having completed this unit, you should be able to:

- Prepare for problems before they occur
- Characterize a problem from its symptoms
- Collect diagnostic data
- Implement a relief or recovery plan
- Troubleshoot a problem

© Copyright IBM Corporation 2013

Figure 3-34. Unit summary

WA5913.0

Notes:

Checkpoint questions

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?

© Copyright IBM Corporation 2013

Figure 3-35. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. The following prevention techniques should be implemented:
 - Implement problem prevention best practices
 - Keep good system documentation
 - Have a diagnostic data collection plan
 - Have a relief or recovery plan
 - Keep a maintenance plan: Scheduled and emergency
2. The following are key steps for problem determination:
 - Preparation before a problem occurs
 - Organizing the investigation
 - Consideration of relief options
 - Initial investigation (phase 1)
 - In-depth investigation (phase 2)
3. Relief options are actions that you can take to address a problem while you are conducting a problem investigation. For example, restart one or more components, or reduce the load.

© Copyright IBM Corporation 2013

Figure 3-36. Checkpoint answers

WA5913.0

Notes:

Unit 4. Gathering diagnostic data

What this unit is about

This unit explains how to use various resources for gathering diagnostic data that includes logs, tracing, and memory dumps. In addition, different tools are described that can be used to analyze the diagnostic data.

What you should be able to do

After completing this unit, you should be able to:

- Identify and access several resources for problem investigation
- Identify, locate, and configure server log files
- Enable HPEL logging for a server and use log viewer tools
- Describe and use cross-component trace (XCT)
- Generate JVM-related diagnostic data by using the administrative console and other tools

How you will check your progress

- Checkpoint questions
- Lab exercise

References

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

Unit objectives

After completing this unit, you should be able to:

- Identify and access several resources for problem investigation
- Identify, locate, and configure server log files
- Enable HPEL logging for a server and use log viewer tools
- Describe and use cross-component trace (XCT)
- Generate JVM-related diagnostic data by using the administrative console and other tools

© Copyright IBM Corporation 2013

Figure 4-1. Unit objectives

WA5913.0

Notes:



Topics

- Resources for problem investigation
- Data gathering tools
- Logs and tracing: Basic mode
- High Performance Extensible Logging (HPEL)
- Gathering JVM-related data

© Copyright IBM Corporation 2013

Figure 4-2. Topics

WA5913.0

Notes:

4.1. Resources for a problem investigation

Resources for a problem investigation



© Copyright IBM Corporation 2013

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

8.0

Figure 4-3. Resources for a problem investigation

WA5913.0

Notes:



WebSphere knowledge base

- The WebSphere knowledge base is composed of:
 - IBM Support Portal
 - DeveloperWorks website
 - WebSphere Application Server Information Center
- Choose key words to research the problem in available knowledge sources:
 - Error codes, exception names, APARs, fix packs, problem determination tools, MustGather documents
- Use the search tool on the WebSphere Application Server support portal
 - Automatically search product information center for error codes

© Copyright IBM Corporation 2013

Figure 4-4. WebSphere knowledge base

WA5913.0

Notes:

WebSphere Application Server Support page:

<http://www.ibm.com/software/webservers/appserv/was/support/>

The next few slides relate to the information-gathering phase of a problem investigation. All problem determination data on the IBM Support site is organized into a set of predefined problem categories or “components”. Categories for WebSphere Application Server include “100% CPU Usage,” “Admin Console,” “Class-loader,” and “Crash.” They are also used to manage most support processes and other problem determination assets. For a list of components and categories, see:
<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, and problem determination tools.
- AlphaWorks provides links to new and emerging tools and technologies. Many problem determination tools can be found here: <http://www.alphaworks.ibm.com/>
- APAR is “authorized program analysis report”; it tracks software defects reported by customers.

Support home

Support for my selected products

Support shortcuts

- WebSphere Application Server
 - Fixes (downloads)
 - Open a new service request - Sign in
 - Product documentation
 - Problem resolution
 - Plan an installation or upgrade

Featured links

- Support registrations
- WebSphere Application Server
 - Support technical exchanges
 - Featured documents
 - End of Support for 6.1 is 30 September 2013
 - Steps to getting support
 - Evaluate: IBM WebSphere Application Server

Flashes and alerts

Alerts: [Latest security bulletins](#)

- WebSphere Application Server
 - 26 Jun 2013: Security Bulletin: WebSphere Application Server Java ...
 - 25 Jun 2013: Security Bulletin: Potential Security Vulnerabilities ...
 - 18 Jun 2013: Security Bulletin: Possible Security Exposure in ...
 - 18 Jun 2013: Security Bulletin: Potential security vulnerability in ...
 - 14 Jun 2013: Security Bulletin: Potential Security Vulnerabilities ...

© Copyright IBM Corporation 2013

Figure 4-5. IBM Support Portal

WA5913.0

Notes:

IBM's support website is called the IBM Support Portal. This site provides a consistent support interface for all IBM products and allows you to customize the site to your specific needs. In this example, the site is customized to show content relevant to WebSphere Application Server:

<http://www.ibm.com/support/entry/portal/Overview>

Troubleshoot

Support for my selected products

Before submitting a request

- ▶ WebSphere Application Server

Featured troubleshooting links

- ▶ WebSphere Application Server
 - Collect troubleshooting data
 - Quicklinks - Resource reference list for WebSphere software products
 - Evaluate: IBM WebSphere Application Server
 - Search results: All APARs
 - View all troubleshooting links

Problem resolution

- ▶ WebSphere Application Server

Flashes and alerts (Troubleshooting)

Alerts: [Latest security bulletins](#)

- ▶ WebSphere Application Server

Troubleshooting top ten

Most recent | Most viewed

- ▶ WebSphere Application Server

Troubleshooting tools

- ▶ WebSphere Application Server

IBM Support Portal Adviser beta

Troubleshooting documentation

- MustGather documents
 - What data to collect for specific problems
- View all APARs
- Tools
- Documentation
 - Troubleshooting guide
 - Information center

© Copyright IBM Corporation 2013

Figure 4-6. WebSphere Support Portal: Troubleshooting

WA5913.0

Notes:

After you select the WebSphere Application Server product, the support pages are focused on that product. As you can see in this screen capture, the Featured troubleshooting links are relevant to WebSphere Application Server. There are links to MustGather documents, APARs, troubleshooting tools, and technical documentation.

MustGather documents aid in problem determination, and they save time in resolving Problem Management Records (PMRs). These documents are on the product support sites and contain specific instructions about what documentation to gather for specific problems.

The information center has extensive documentation on troubleshooting.

© Copyright IBM Corporation 2013

Figure 4-7. Information center: Troubleshooting and support

WA5913.0

Notes:

The information center is a good resource for troubleshooting. Specific problem areas are documented, and a search facility is provided. Numerous articles describe how to troubleshoot WebSphere Application Server components:

- Messaging
- Security
- Transactions
- Web services
- Many more

MustGather example (1 of 3)

MustGather: Installation Manager issues for installing and updating WebSphere Application Server V8.0

This article describes the procedure for collecting data for when installation or updates of WebSphere Application Server V8.0 using IBM Installation Manager does not complete successfully. Gathering this information before calling IBM support will help familiarize you with the troubleshooting process and save time analyzing the data.

Collecting data using Installation Manager

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

Follow the directions in this MustGather if you are experiencing issues installing WebSphere Application Server V8.0 for AIX, HP-UX, Linux, Solaris, Windows, or z/OS. You can choose one of the topics listed below, or simply scroll further down this page.

- [Collecting data using the Installation Manager graphical \(GUI\) interface](#)
- [Collecting data using the Installation Manager command line \(imcl\) interface](#)
- [Collecting data for Installation Manager on z/OS](#)
- [Collecting manually when Installation Manager is unable to automatically gather data](#)

© Copyright IBM Corporation 2013

Figure 4-8. MustGather example (1 of 3)

WA5913.0

Notes:

MustGather documents can be accessed from within IBM Support Assistant or on the IBM Support website.

The screen capture displays a MustGather document for Installation Manager issues for installing and updating WebSphere Application Server V8.0. This page provides several links that give details on how to collect data for the Installation Manager.

MustGather example (2 of 3)

Collecting manually when Installation Manager is unable to automatically gather data

If Installation Manager is experiencing a problem where it is unable to gather the files using the methods described above, then click the section header below, titled "Manually collecting files". Follow those directions which explain how to locate and collect individual files.

If Installation Manager is functional and it is able to collect the data, then you do not need to follow those directions; instead, use one of the automated methods described above.

Optional additional data

The installer writes some additional data which can be helpful for verifying whether an installation is complete. We can typically proceed with problem analysis without this data, so we consider it optional. We have included it in this mustgather for completeness.

- Collect these two files:
WAS_HOME/properties/version/installed.xml
WAS_HOME/properties/version/history.xml
- Collect the .was.installocation.registry file. The location varies, depending on the user running Installation Manager and the operating system:
 - Administrator or root user
 - AIX /usr/.ibm
 - HP-UX, Linux, /opt/.ibm
 - Solaris
 - Windows Windows directory
 - Non-administrator or non-root user
 - AIX, HP-UX, Linux, <user_home>/.ibm
 - Solaris
 - Windows <user_home>\.ibm

© Copyright IBM Corporation 2013

Figure 4-9. MustGather example (2 of 3)

WA5913.0

Notes:

This screen capture shows the page that is displayed when you click the link on the previous page for "Collecting manually when Installation Manager is unable to automatically gather data."

MustGather example (3 of 3)

- Collecting data tab > MustGather 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 \$AdminControl 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 `server.xml` files for both nodeagent and dmgr:
 - i. Enable discovery tracing on the dmgr:
 1. Edit the `server.xml` file for the dmgr located in the following directory:

`ND_PROFILE/config/cells/cellname/nodes/nodename/servers/dmgr`

© Copyright IBM Corporation 2013

Figure 4-10. MustGather example (3 of 3)

WA5913.0

Notes:

On the IBM Support site, many MustGather documents now have companion “troubleshooting” and “analyzing data” documents (see, for example, <http://www.ibm.com/support/docview.wss?uid=swg21145599>).

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

Most of MustGather documents for WebSphere Application Server now have a corresponding AutoPD script (in IBM Support Assistant). So, you can either follow the steps from the MustGather document by hand, or run the AutoPD script, which does the work more or less automatically.

Using the IBM Support Assistant Data Collector (1 of 2)

- The IBM Support Assistant Data Collector for WebSphere Application Server is a tool for gathering diagnostic data from an application server
 - It replaces the collector tool, which is deprecated
- Also provides symptom analysis support for the various categories of problems
- The tool runs in console mode by starting the launch script from the command line
- In a Windows environment, run the
`<profile_root>/bin/isadc.bat` command
- In a Linux, AIX, HP-UX, Solaris, IBM i, or z/OS environment, run the
`<profile_root>/bin/isadc.sh` command

© Copyright IBM Corporation 2013

Figure 4-11. Using the IBM Support Assistant Data Collector (1 of 2)

WA5913.0

Notes:

The IBM Support Assistant Data Collector for WebSphere Application Server tool focuses on automatic collection of problem data. It also provides symptom analysis support for the various categories of problems that IBM software products encounter. Information pertinent to a type of problem is collected to help identify the origin of the problem under investigation. The tool assists customers by reducing the amount of time it takes to reproduce a problem with the correct RAS tracing levels set. It also reduces the effort that is required to send the appropriate log information to IBM Support.

Using the IBM Support Assistant Data Collector (2 of 2)

- Diagnostic data is gathered as specified in the MustGather document for each problem type
- Some options have subtypes
- Security...has subtypes
 - SSL
 - Java 2
 - Others

```

Terminal
File Edit View Terminal Tabs Help
was85host:/opt/IBM/WebSphere/AppServer/profiles/profile1/bin # ./isadc.sh
Starting IBM Support Assistant Data Collector in console mode...

ApplicationServer: 2.0.2.20120316
IBM Support Assistant Data Collector:
2.0.1.GA20120316-1743
Common Inventory Sub Agent: 6.3.0.20120123

Create the collection zip file at /root/localhostNode01Cell-
was85host01Node01-WAS-ISADC.zip?
[1] Yes
[2] No
> 1

Enter the number of the IBM Support Assistant Data Collector option you
want to execute.
[1] Default Collection [ distributed, IBM i, z/OS ]
[2] General... [ distributed, IBM i ]
[3] Administration... [ distributed, IBM i ]
[4] Security... [ distributed, IBM i ]
[5] Connectors... [ distributed, IBM i ]
[6] Containers... [ distributed, IBM i ]
[7] JDK... [ distributed ]
[8] Runtime... [ distributed, IBM i ]
[9] HTTP... [ distributed, IBM i ]
[10] Service Oriented Architecture... [ distributed, IBM i ]
[11] Intelligent Management Pack... [ distributed, IBM i ]
[12] Quit

```

© Copyright IBM Corporation 2013

Figure 4-12. Using the IBM Support Assistant Data Collector (2 of 2)

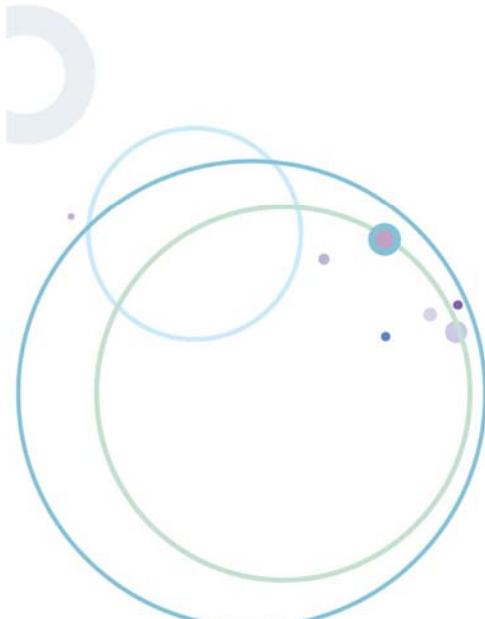
WA5913.0

Notes:

Run the IBM Support Assistant Data Collector tool with the user ID for which you configured your WebSphere Server instance. Depending on what collection you are running, you are asked more questions to complete the data collection activities. A script might require more configuration information, information about the sequence of events that lead up to the problem you are dealing with, or your preferences on how it completes the collection. At each step, the choices are presented as numbered lists, and you input the number of your selection and press the Enter key. When input is required, prompts are displayed at which you enter your response and press the Enter key. You can find collection details for each WebSphere Application Server problem type in their corresponding MustGather documents.

4.2. Data gathering tools

Data gathering tools



© Copyright IBM Corporation 2013

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

8.0

Figure 4-13. Data gathering tools

WA5913.0

Notes:

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

- Logging and tracing (Basic and HPEL modes)
- Troubleshooting panels in the administrative console
- Specialized tracing and runtime checks:
 - Connection leak detection
 - Memory leak detection
 - Enabled by tracing a specific component or setting a specialized custom property
- First-failure data capture (FFDC)
 - Always enabled
 - 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

© Copyright IBM Corporation 2013

Figure 4-14. Types of tools and where to find them (1 of 4)

WA5913.0

Notes:

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

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

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

- Captured data, which is saved in log files for use in analysis
- An index file that references all of the exceptions that FFDC logs
- 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 (afterward, it is deleted). Retrieve these log files with 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 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 number of days between purging the FFDC log files. The value of the `ExceptionFileMaximumAge` property must be a positive number.

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

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

- JVM diagnostic data can be generated per server from the administrative console
 - Verbose garbage collection
 - Heap memory dump, javacore, system memory dump
- Performance-related tools:
 - Performance Monitoring Infrastructure (PMI) is a facility specific to WebSphere
 - Tivoli Performance Viewer, available from the administrative console, is the primary tool for viewing PMI data
 - Java Health Center: GC and memory monitoring
 - Request metrics can be accessed by using Application Response Measurement (ARM) infrastructure
 - IBM Application Performance Diagnostics Lite is a powerful, lightweight tool that can be used to optimize the performance of Java EE applications

© Copyright IBM Corporation 2013

Figure 4-15. Types of tools and where to find them (2 of 4)

WA5913.0

Notes:

The Java Diagnostics Guides also provide details about some of these tools and how to generate various types of memory dumps. You can find the V6 guide here:

<http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/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.

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

For more information about IBM Application Performance Diagnostics Lite, see the developerWorks website: <http://www.ibm.com/developerworks/servicemanagement/apm/apd/>

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

Monitoring and detection:

- **Hung thread detection facility** is directly connected into the WebSphere Application Server run time
- **Performance and diagnostic advisors** are accessible from the administrative console
- Performance advisor is available in the Tivoli Performance Viewer

Specific component investigation:

- **System Management Configuration Validation** can detect errors in the XML configuration files
 - It is accessible from the administrative console
- **DumpNameSpace** dumps the contents of the JNDI name space at a particular server
 - It is a stand-alone product with WebSphere Application Server (<install_root>/bin)
- **Class loader viewer** helps resolve class loading issues and is accessible through the troubleshooting menu of the administrative console

© Copyright IBM Corporation 2013

Figure 4-16. Types of tools and where to find them (3 of 4)

WA5913.0

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 do 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 because of many runtime safety checks, can still crop up. These errors can be because of as-yet-undiscovered product defects, unexpected events that occur during configuration operations (like crashes), or operator mistakes during configuration. This facility is embedded inside the WebSphere Application Server runtime itself, and can be started from the administration console (in the troubleshooting panel).

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

- Installation issues:
 - **Installation Manager**
 - **Installation verification tool (IVT)** verifies each profile from its First Steps console with the IVT tool (<profile_root>/bin)
 - **Installation verification utility (installver)**: Replaced by the verification capabilities of the Installation Manager in version 8
 - **VersionInfo, HistoryInfo, GenHistoryReport** is used to track changes and apply fix packs; all bundled with WebSphere Application Server
- Debuggers and profilers are valuable to application support, but are outside the scope of this course
 - Rational Application Developer
 - IBM Assembly and Deploy Tools
 - Java Health Center can be used for profiling

© Copyright IBM Corporation 2013

Figure 4-17. Types of tools and where to find them (4 of 4)

WA5913.0

Notes:

WebSphere Application Server Version 7 and earlier had an installation verification utility, the installer command, that would verify checksums of installed files against a bill of materials that was included with the product. In WebSphere Application Server Version 8.0 and later, where the installation is based on the Installation Manager rather than on InstallShield MultiPlatform (ISMP), the verification capabilities of the Installation Manager replace the installer command.

Debuggers and profilers:

- The ability to debug and profile is also often valuable to the application support process. WebSphere Application Server provides these facilities through the JVM, either through the JVMPPI 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 IBM Assembly and Deploy Tools, include a powerful debugger and profiler tool that connects to these facilities. For profiling-related work, you might also consider the Performance Inspector family of tools, which provides various tools to extract and analyze runtime

performance information from a JVM, by using these same basic interfaces. These tools are available on alphaWorks (for Windows) or SourceForge (for Linux).

4.3. Logs and tracing: Basic mode

Logs and tracing: Basic mode



© Copyright IBM Corporation 2013

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

8.0

Figure 4-18. Logs and tracing: Basic mode

WA5913.0

Notes:



Basic mode logs and tracing

- Logs:
 - Report key events in the system and enabled by default
 - Incur minimal performance usage
 - Look for key messages like warnings, errors, and exceptions
- Tracing:
 - Application code-level events; level of detail can be configured
 - Detailed tracing can be enabled by specifying a trace string
 - Trace helps you understand the flow of the application
 - HPEL consolidates key messages on a particular node and contains extended service information
- JMX-based monitoring:
 - Most key events are exported as JMX events
 - Tools can be built for remotely monitoring and capturing events

© Copyright IBM Corporation 2013

Figure 4-19. Basic mode logs and tracing

WA5913.0

Notes:

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

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

Tracing is explained later in this unit.

Log files and locations

Log files	Location
SystemOut.log	<was_root>/profiles/<profile_name>/logs/<server_name>
SystemError.log	<was_root>/profiles/<profile_name>/logs/<server_name>
startServer.log	<was_root>/profiles/<profile_name>/logs/<server_name>
stopServer.log	<was_root>/profiles/<profile_name>/logs/<server_name>
activity.log	<was_root>/profiles/<profile_name>/logs/<server_name>
trace.log	<was_root>/profiles/<profile_name>/logs/<server_name>
http_plugin.log	<plugin_root>/logs/<webserver_name>
native_stdout.log	<was_root>/profiles/<profile_name>/logs/<server_name>
native_stderr.log	<was_root>/profiles/<profile_name>/logs/<server_name>
btrace.1	<was_root>/profiles/<profile_name>/logs/<server_name>

Note: btrace.1 is used for the binary trace facility of Intelligent Management, which is covered in a later unit

© Copyright IBM Corporation 2013

Figure 4-20. Log files and locations

WA5913.0

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 and also user-program information (sent by a System.out.xxx code in the program).

The startServer.log and stopServer.log files can also be found under the <was_root>\logs\<servername> directory; they contain information that the server logs 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.

There are several 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 that are 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 that runs 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 by using the `print()` and `println()` methods that the streams define. If there is 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 that runs 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. As with JVM logs, there is a set of process logs for each application server since each JVM is an operating system process. For a WebSphere Application Server Network Deployment configuration, there is a set of process logs for the deployment manager and each node agent.
- IBM service logs: The IBM service logs contain the WebSphere Application Server messages that are written to the `System.out` stream. Some special messages that contain extended service information that is normally not of interest, but can be important when analyzing problems, are also written. 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 more diagnostic capabilities. In addition, the binary format provides capabilities that IBM Support organizations use. The HTTP server plug-in log is covered later in this presentation.



Configuring JVM logs

- Click: **Troubleshooting > Logs and trace > *server_name* > JVM Logs**
- System.out and System.err logs can be configured from this page
- Logs are self-managing
 - Roll-over based on time or file size
 - Number of historical log files is configurable
- Use the **Runtime** tab to view the logs

The screenshot shows the 'Runtime' tab selected in the top navigation bar. Under 'General Properties', the 'System.out' section is active. The 'File Name' field contains the value \${SERVER_LOG_ROOT}/SystemOut.log, which is highlighted with a yellow background. Below it, 'File Formatting' is set to 'Basic (Compatible)'. The 'Log File Rotation' section includes options for 'File Size' (checked, maximum size 1 MB) and 'Time' (unchecked, start time 24, repeat time 24 hours). A note below says 'Maximum Number of Historical Log Files. Number in range 1 through 200.' with the value 1 entered. In the 'Installed Application Output' section, 'Show application print statements' is checked, and 'Format print statements' is also checked.

© Copyright IBM Corporation 2013

Figure 4-21. Configuring JVM logs

WA5913.0

Notes:

You can also configure the logs from this alternative method: **Servers > Application servers > *server_name* > Logging and Tracing > JVM Logs**.

Viewing runtime messages in the console

The screenshot shows the 'Runtime Events' interface. At the top, there's a message stating that runtime events are disabled by default ('None'). A dropdown menu is set to 'Info' and an 'Apply' button is below it. Below this is a 'Preferences' section with a toolbar containing icons for sorting and filtering. A table lists three runtime events:

Timestamp	Message Originator	Message
Apr 5, 2012 3:04:17 PM EDT	com.ibm.ws.ssl.core.WSX509TrustManager	CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with
Apr 11, 2012 10:07:07 AM EDT	com.ibm.ws.webcontainer	SRVE0255E: A WebGroup/Virtual Host to handle /favi
Apr 11, 2012 10:06:56 AM EDT	com.ibm.ws.webcontainer	SRVE0255E: A WebGroup/Virtual Host to handle /favi

Total 3

© Copyright IBM Corporation 2013

Figure 4-22. Viewing runtime messages in the console

WA5913.0

Notes:

Navigate to **Troubleshooting > Runtime Messages > Runtime error**.

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: you might have multiple pages of messages, and the button on the bottom of the page allows you to see them all. Information is displayed on the detail screen for the event so you can resolve the problem with user action.



HTTP plug-in logs and tracing

- Click **Servers > Server Types > Web servers > web_server_name > Plug-in properties > Configuration tab > Plug-in logging** to configure plug-in logs and tracing
- 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: produces much output)



- Use the tool **IBM Web Server Plug-in Analyzer for WebSphere Application Server** to analyze the plug-in trace
 - Available in the IBM Support Assistant

© Copyright IBM Corporation 2013

Figure 4-23. HTTP plug-in logs and tracing

WA5913.0

Notes:

IBM Web Server Plug-in Analyzer for WebSphere Application Server helps discover potential problems with trace and configuration files during use of WebSphere Application Server. The tool parses both plug-in configuration and corresponding trace files and then applies pattern recognition algorithms to alert users to possible inconsistencies.

The tool provides a list of HTTP return codes, URIs, graphical presentations of available clusters, and server topologies from the configuration and trace files. For more information, see:
<http://www.alphaworks.ibm.com/tech/wspa>

IBM Web Server Plug-in Analyzer for WebSphere Application Server is available as a tool add-on in the IBM Support Assistant.

Configuration

General Properties

Enable logging service at server start-up

NCSA Access logging

Enable access logging

* Access log file path
\${SERVER_LOG_ROOT}/http_access.log

Access log maximum size
500 MB

Maximum number of historical files
1

* NCSA access log format
Common

Error logging

Enable error logging

* Error log file path
\${SERVER_LOG_ROOT}/http_error.log

Error log maximum size
500 MB

Maximum number of historical files
1

* Error logging level
Warning

Figure 4-24. Embedded HTTP server logs

WA5913.0

Notes:

Embedded HTTP server logs:

- Administrative console panels for configuring embedded HTTP server logs (access and error).
- From main application server panel, navigate to **Troubleshooting > server_name > HTTP Error and NCSA Access Logging**.
- Access and error logs can be controlled separately.
- When maximum file size is reached, the oldest entries are removed.

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. There is not a runtime tab for these logs. Logging begins after changes are saved to your configuration and the application server restarts.

Diagnostic traces

- Trace files show the time and sequence of methods that are called from WebSphere base classes
 - Use trace data to help pinpoint the failure
- Trace can be started:
 - While the server is running, use Runtime Diagnostic Trace
 - When the server is started, use Configuration Diagnostic Trace
- Trace output can be directed to:
 - Memory ring buffer that is 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

© Copyright IBM Corporation 2013

Figure 4-25. Diagnostic traces

WA5913.0

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:

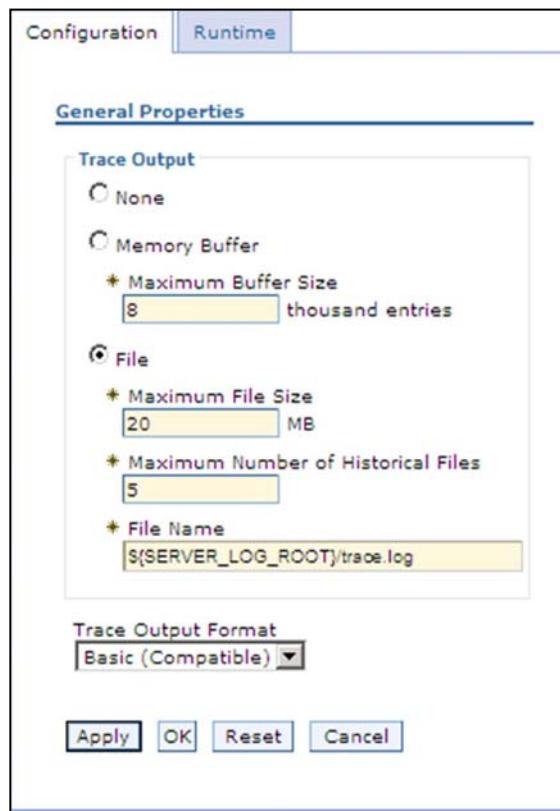
1. Configure an output destination to which trace data is sent.
2. Enable trace for the appropriate WebSphere Application Server or application components.
3. Run the application or operation to generate the trace data.

4. Analyze the trace data or forward it to the appropriate organization for analysis.
5. Click **Troubleshooting > Logs and Trace > *server_name*** in the administrative console.



Enabling diagnostic trace

- Troubleshooting > Logs and Trace > *server_name* > Diagnostic Trace
- Configure Trace Output
 - None
 - Memory buffer
 - File (default)
- Configure Trace Output Format
 - Basic (recommended by IBM support)
 - Advanced
- **Note:** Configure Log Detail Level to get trace output



© Copyright IBM Corporation 2013

Figure 4-26. Enabling diagnostic trace

WA5913.0

Notes:

Specifying the trace settings under the Configuration tab requires a server restart for the tracing to take effect. However, enabling the tracing under the Runtime tab takes effect immediately (which can be 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 report only the current system information; if the server is already having issues, enabling runtime traces does not report what caused the problem to begin with, only why current actions fail. Therefore, IBM Support suggests 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 that is captured does not get wrapped and lost. By default, these values are set to 20 MB for the file size, and one backup file. As a result, when the `trace.log` file reaches 20 MB in size, a new `trace.log` gets created. The previous `trace.log` file is renamed to `trace_<date>.log` where `<date>` is the date and time of the last entry in the trace. However, as soon as the new `trace.log` file reaches 20 MB in size again, it then wraps into a new trace file as before, but the previous `trace_<date>.log` file is deleted. Therefore, it is important to ensure that the Number of Historical Files and Maximum File Size are large enough when re-creating 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.



Diagnostic trace dump and runtime

Configuration **Runtime**

General Properties

Save runtime changes to configuration as well

Trace Output

Memory Buffer

Maximum Buffer Size
[] thousand entries

Dump File Name
[]

Dump

File

Maximum File Size
20 MB

Maximum Number of Historical Files
1

File Name
\${SERVER_LOG_ROOT}/trace.log

View

Buttons: Apply | OK | Reset | Cancel

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

© Copyright IBM Corporation 2013

Figure 4-27. Diagnostic trace dump and runtime

WA5913.0

Notes:

The trace analyzer for WebSphere Application Server enables you to view a complex text-based trace file in a more easy-to-use GUI interface.

Setting the log detail level

- Logs and trace > `server_name` > Change Log Detail Level
- Log detail level affects tracing *and* regular logging
 - Setting levels below `info` reduces the amount of data in logs
 - `*=off` disables logging altogether
- Trace levels (`fine`, `finer`, `finest`) do not appear in the trace file unless logging is enabled
- Use the graphical menu to type in or set the log string
- Default is `*=info`
- User-created applications can be instrumented too, and be included in the trace output

Figure 4-28. Setting the log detail level

WA5913.0

Notes:

Log levels control the events that Java logging processes.

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 consists of events at levels `fine`, `finer`, and `finest`, can be written only to the trace log. Therefore, if you do not enable diagnostic trace, setting the log detail level to `fine`, `finer`, or `finest` does not affect the data that is logged.

Log string syntax: `<component / group> = <log level>`

Examples include:

- `com.ibm.ws.classloader.ClassGraph=fine`

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

- `EJBContainer=fine`

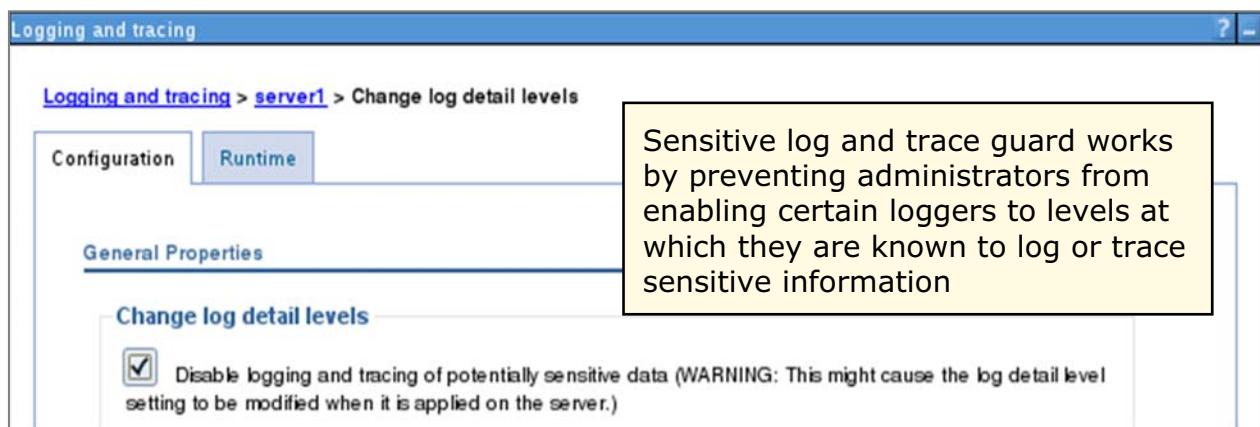
Enables the least verbose trace level for all components in the EJBContainer group

- `com.ibm.ws.classloader.*=finer`
Enables detailed trace for all classes in the `com.ibm.ws.classloader` package
- `*=info` Sets
Sets the log level for all components to info (default: no trace output)



Enabling and disabling sensitive log and trace guard

- Administrators can prevent sensitive information, such as data provided from users in HTTP requests, from being written in log and trace files
- In some cases, if access to private data can help with debugging, you can disable sensitive log and trace guard
 - For example, you might see that a credit card number that was entered in a web form did not have the required number of digits



© Copyright IBM Corporation 2013

Figure 4-29. Enabling and disabling sensitive log and trace guard

WA5913.0

Notes:

Administrators that use WebSphere Application Server can prevent sensitive information, such as data that users provide in HTTP requests, from being written in log and trace files. In some cases, when having access to private data can help with debugging, you might want to disable sensitive log and trace guard. For example, you might see that a credit card number that was entered in a web form did not have the required number of digits.

Sensitive log and trace guard works by preventing administrators from enabling certain loggers to levels at which they are known to log or trace sensitive information.

After you enable sensitive log and trace guard, the server is now configured to prevent known sensitive loggers from writing sensitive content to the log and trace files. After you disable sensitive log and trace guard, the server is now configured to allow known sensitive loggers to write sensitive content to the log and trace files. If you used the deployment manager to complete these steps, you might be required to synchronize the node agent on the target node before restarting the server.

See the information center topic "Enabling and disabling sensitive log and trace guard" at:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/ae/welcome_ndmp.html

Enabling trace by using Jython scripts (1 of 2)

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

```

cellName = 'myCell'
nodeName = 'myNode'
serverName = 'myServer'

server = AdminConfig.getId(
    '/Cell:%(cellName)s/Node:%(nodeName)s/Server:%(serverName)s' %
    locals() )
ts = AdminConfig.list( 'TraceService', server )

# print AdminConfig.showAttribute( ts, 'startupTraceSpecification' )

AdminConfig.modify( ts, [ [ 'startupTraceSpecification',
    'com.ibm.ws.*=all=enabled' ] ] )
AdminConfig.save()

```

© Copyright IBM Corporation 2013

Figure 4-30. Enabling trace by using Jython scripts (1 of 2)

WA5913.0

Notes:

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

Enabling trace by using Jython scripts (2 of 2)

- To enable runtime traces with wsadmin, use these commands:

```
ts = AdminControl.queryNames(  
    'type=TraceService,node=%(nodeName)s,process=%(serverName)s,*' %  
    locals() )  
# print AdminControl.getAttribute( ts, 'traceSpecification' )  
AdminControl.setAttribute( ts, 'traceSpecification',  
    'com.ibm.ws.*=all=enabled' )
```

- You can find trace strings for WebSphere components that are associated with a specific problem in the MustGather documents

© Copyright IBM Corporation 2013

Figure 4-31. Enabling trace by using Jython scripts (2 of 2)

WA5913.0

Notes:

JVM log and trace log headers

```
***** Start Display Current Environment *****
1 WebSphere Platform 8.0.0.2 [ND 8.0.0.2 cf021148.03] running with process name
was8host01Cell01\was8host01Node01\server1 and process id 5828
Host Operating System is Windows XP, version 5.1
2 Java version = 1.6.0, Java Compiler = j9jit26, Java VM name = IBM J9 VM
was.install.root = C:\Program Files\IBM\WebSphere\AppServer
user.install.root = C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1
Java Home = C:\Program Files\IBM\WebSphere\AppServer\java\jre
ws.ext.dirs = C:\Program Files\IBM\WebSphere\AppServer\java\lib;C:\Program
Files\IBM\WebSphere\AppServer\profiles\profile1\classes;
...<Truncated>
3 Classpath = C:\Program
Files\IBM\WebSphere\AppServer\profiles\profile1\properties;C:\Program
Files\IBM\WebSphere\AppServer\properties;
...<Truncated>
Java Library path = C:\Program
Files\IBM\WebSphere\AppServer\lib\native\win\x86_32;/C:\Program
Files\IBM\WebSphere\AppServer\java\jre\bin\default;
....< Truncated>
4 Orb Version = IBM Java ORB build orb626-20111107.00
Current trace specification = *=info:WAS.j2c=all
***** End Display Current Environment *****
```

1. WebSphere level
2. Java version
3. Classpath
4. Trace spec

© Copyright IBM Corporation 2013

Figure 4-32. JVM log and trace log headers

WA5913.0

Notes:

This header is also produced each time that the server is started. From this header, you can see the exact:

1. WebSphere Application Server level
2. Java version
3. 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, and also the WebSphere Application Server variables `was.install.root` and `user.install.root`. During startup, the trace specification in use is printed right after the header.

Current trace specification: *=info:WAS.j2c=all

Default trace format

- **Basic format:** the following format is used to record trace events:

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

- **Example:**

```
[8/3/11 16:41:28:041 EDT] 00000020 CacheHook      3 Request Type:GET
[8/3/11 16:41:28:041 EDT] 00000020 CacheHook      3 dncResult: true
    skipCache: false
[8/3/11 16:41:28:041 EDT] 00000020 FragmentCompo 3 starting fragment
    composer for /secure/layouts/contentLayout.jsp
    parent=/com.ibm.ws.console.events.forwardCmd.do
[8/3/11 16:41:28:041 EDT] 00000020 ESISupport     >
    handleESIPreProcessing() Entry
```

© Copyright IBM Corporation 2013

Figure 4-33. Default trace format

WA5913.0

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 Aug. 3, 2011 at 16:41:28:041 EDT time.
- The trace output shows a single threaded process with ID 000000a.
- The class name that is executing is “CacheHook.”

If the class name is over 13 characters, only the first 13 characters are displayed.

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

```
[8/3/11 16:41:28:041 EDT] 00000020 CacheHook      3 Request Type:GET
```

- Debug point in the CacheHook class.

```
[8/3/11 16:41:28:041 EDT] 00000020 ESISupport     > handleESIPreProcessing() Entry
```

- The `handleESIPreProcessing()` method on `ESISupport` is executing.

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

Reading a log or trace file (1 of 2)

- Example log record format

```
[4/21/12 10:10:45:134 EDT] 00000001 WSKeyStore W
CWPKI0041W: One or more key stores are using the default
password.
```

- Time stamp = [4/21/12 10:10:45:134 EDT]
- Thread ID = 00000001
- Logger = WSKeyStore
- Message type = W
- Message code = CWPKI0041W
- Message = One or more key stores are using the default password.

Msg type	Description
1,2,3	Trace info: fine, finer, finest
A	Audit
W	Warning
Z	Type was not recognized
E	Error
D	Detail
C	Configuration
F	Fatal (exits process)
I	Information
O	Program output (system.out)
R	Program output (system.err)

© Copyright IBM Corporation 2013

Figure 4-34. Reading a log or trace file (1 of 2)

WA5913.0

Notes:

The following list defines the sections of a trace entry:

- TimeStamp:** The time stamp is formatted to agree with the locale of the process where it is formatted. It includes a fully qualified date (YYMMDD), 24-hour time with millisecond precision, and the time zone.
- ThreadId:** An eight-character hexadecimal value that is generated from the hash code of the thread that issued the trace event.
- ThreadName:** The name of the Java thread that issued the message or trace event.
- ShortName:** The abbreviated name of the logging component that issued the trace event. This abbreviated name is typically the class name for WebSphere Application Server internal components, but might be some other identifier for user applications.
- LongName:** The full name of the logging component that issued the trace event. This name is typically the fully qualified class name for WebSphere Application Server internal components, but might be some other identifier for user applications.
- EventType:** A one-character field that indicates the type of the trace event. Trace types are in lowercase.

- **ClassName:** The class that issued the message or trace event.
- **MethodName:** The method that issued the message or trace event.
- **CWPKI0041W:** One or more keystores are using the default password.
 - **Explanation:** When the Application Server starts for the first time as a stand-alone application server or in a Network Deployment configuration, each server creates a keystore and truststore for the default Secure Sockets Layer (SSL) configuration. When the application server creates these files, by default, it uses WebAS for the password. Do not use the default password in production. The warning message suggests that you change the password.
 - **Action:** To eliminate this warning message, use the administrative console to change the default password for the keystore and the truststore, and also change these passwords by editing the `ssl.client.props` file. When you change the passwords in the `ssl.client.props` file, you must use the PropFilePasswordEncoder utility to re-encode the new passwords.

Reading a log or trace file (2 of 2)

- Timestamps provide good clues:
 - Timestamps are real system time values
 - Useful when comparing traces from different processes and correlating events of different servers
- Look for exceptions (search for exception from top of stack trace)
 - Events before exception are probable causes
 - Events after exception are recovery attempts
- Useful to follow a single thread
 - Use the Thread ID to gather related messages

© Copyright IBM Corporation 2013

Figure 4-35. Reading a log or trace file (2 of 2)

WA5913.0

Notes:

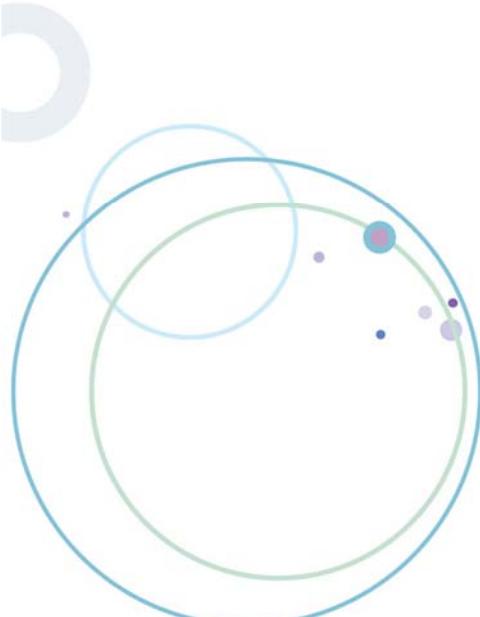
While it is possible to read logs and trace files by hand by using only a text editor, it is suggested that you use a tool such as Log Analyzer, which is available in the IBM Support Assistant.

The following article gives a good introduction to interpreting trace data:

http://www.ibm.com/developerworks/websphere/techjournal/0704_supauth/0704_supauth.html

4.4. High Performance Extensible Logging (HPEL)

High Performance Extensible Logging (HPEL)



© Copyright IBM Corporation 2013

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

8.0

Figure 4-36. High Performance Extensible Logging (HPEL) (1 of 3)

WA5913.0

Notes:

High Performance Extensible Logging (HPEL)

- Performance
 - Log and trace events are stored as binary data so performance is substantially faster than default log and trace framework (Basic mode)
 - Less impact to systems when trace is enabled
- Extensions
 - HPEL includes information about which application each log or trace record is from
 - When used with cross-component trace, HPEL includes information about which request each log or trace record is from
 - Developers can add their own log or trace record extensions
 - HPEL `logViewer` command enables filtering logs or trace data by extension name and value
- Filtering
 - HPEL `logViewer` command helps you filter logs and trace data by date, time, level, thread ID, and so forth

© Copyright IBM Corporation 2013

Figure 4-37. High Performance Extensible Logging (HPEL) (2 of 3)

WA5913.0

Notes:

A number of factors contribute to the overall performance of HPEL logging and tracing.

Log and trace events are each stored in only one place.

Log events, `System.out`, and `System.err` are stored in the log data repository. Trace events are stored in the trace data repository. If the text log file is disabled, HPEL might write log and trace content only to these repositories. Storing each type of event in one place ensures that performance is not wasted on redundant data storage.

Log events, and optionally trace events, are written to the text log file when it is enabled. Since this data is always also stored in the log data and trace data repositories, the text log file content is redundant. The text log is convenient for users who do not want to run the `LogViewer` command-line tool to see their logs and trace; but you can disable the text log if this convenience is not needed.

Data is not formatted unless it is needed.

Formatting data for a user to read uses processor time. Rather than format log event and trace event data at run time, HPEL log and trace data is stored more rapidly in a proprietary binary representation. This fast binary storage improves the performance of the log and trace facility. By

deferring log and trace formatting until the LogViewer is run, sections of the log or trace that are never viewed are never formatted.

Log and trace data is buffered before being written to disk.

High Performance Extensible Logging (HPEL)

- Enable HPEL on any server in the cell
 - Deployment manager
 - Node agent
 - Application server
- Click **Troubleshooting > Logs and trace > server_name**
 - Click **Switch to HPEL Mode**
- Also, available in stand-alone administrative console

Logging and tracing

Logging and tracing > server1

It is recommended that you switch to High Performance Extensible Logging (HPEL) if you have no existing procedures that prevent you from taking advantage of it.

Switch to HPEL Mode (Advised for most installations)

Use this page to select a system log to configure, or to specify a log detail level for components and groups of components. Use log levels to control which events are processed by Java logging.

General Properties

- [Diagnostic Trace](#)
- [JVM Logs](#)
- [Process Logs](#)
- [IBM Service Logs](#)
- [Change log detail levels](#)
- [NCSA access and HTTP error logging](#)

© Copyright IBM Corporation 2013

Figure 4-38. High Performance Extensible Logging (HPEL) (3 of 3)

WA5913.0

Notes:

As soon as the log level is switched to HPEL mode for a server, new links appear in the General Properties section for configuring HPEL logging and tracing. These links are shown on the next slide. A new link appears to change log and trace mode, which allows you to switch back to basic logging.



HPEL logging and tracing configuration

- Use this page to configure HPEL logging and tracing

Logging and tracing

[Logging and tracing > server1](#)

General Properties

[Configure HPEL logging](#)

Directory	C:\WebSphere\AppServer\profiles\profile1/logs/server1
For cleanup, delete records older than	Disabled
For cleanup, maximum size of logs	50 Megabytes

[Configure HPEL trace](#)

Directory	C:\WebSphere\AppServer\profiles\profile1/logs/server1
For cleanup, delete records older than	Disabled
For cleanup, maximum size of trace	50 Megabytes

[Configure HPEL text log](#)

Current status:	Enabled
Directory	C:\WebSphere\AppServer\profiles\profile1/logs/server1
For cleanup, delete records older than	Disabled
For cleanup, maximum size of text log	50 Megabytes

© Copyright IBM Corporation 2013

Figure 4-39. HPEL logging and tracing configuration

WA5913.0

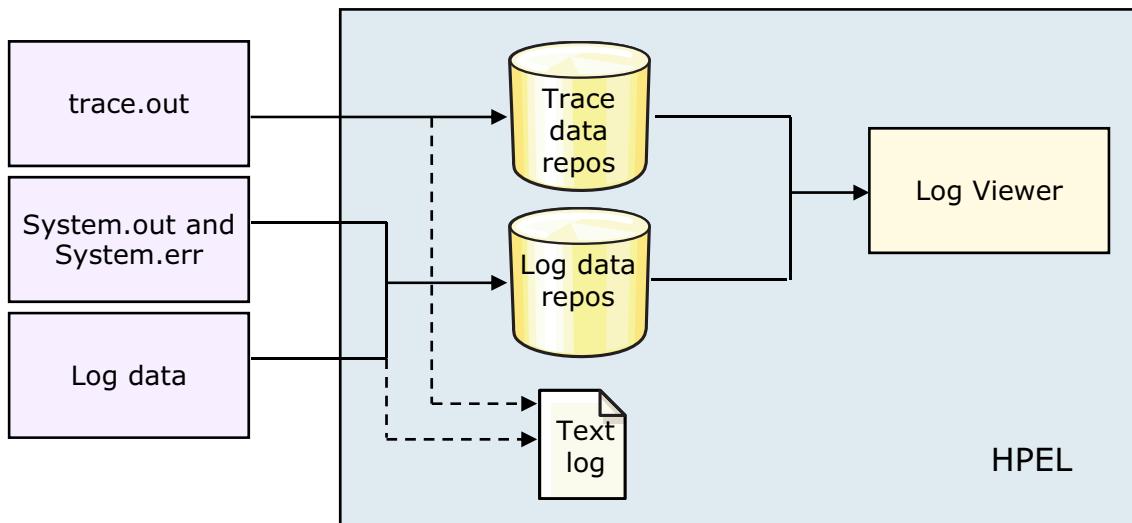
Notes:

This screen capture shows the default configuration for HPEL logging and tracing. To modify the configuration, click any of the links on this page.

After HPEL is enabled for a server, it is good to disable the HPEL text logging. Disabling text logging improves server performance.

HPEL log and trace storage

- HPEL provides a binary log data repository, a binary trace data repository, and a text log file
- When HPEL is enabled and configured, the user is no longer required to understand the details of the individual files to do analysis



© Copyright IBM Corporation 2013

Figure 4-40. HPEL log and trace storage

WA5913.0

Notes:

The user is no longer required to be aware of whether data is written to `SystemOut.log`, `SystemErr.log`, or `trace.log`. With HPEL, the logs are consolidated and the log viewer can be used to view all the data or it can be filtered according to what subset of messages are needed.

The system still uses process logs, `native_stdout.log`, and `native_stderr.log`, and they are found in the default server logs directory.

They are not viewable with the HPEL Log Viewer.

Log files and locations for HPEL

Log files	Location
Logdata (directory)	<was_root>/profiles/<profile_name>/logs/<server_name>
Tracedata (directory)	<was_root>/profiles/<profile_name>/logs/<server_name>
TextLog.log	<was_root>/profiles/<profile_name>/logs/<server_name>
startServer.log	<was_root>/profiles/<profile_name>/logs/<server_name>
stopServer.log	<was_root>/profiles/<profile_name>/logs/<server_name>
native_stdout.log	<was_root>/profiles/<profile_name>/logs/<server_name>
native_stderr.log	<was_root>/profiles/<profile_name>/logs/<server_name>
btrace.1	<was_root>/profiles/<profile_name>/logs/<server_name>

Note: btrace.1 is used for the binary trace facility of Intelligent Management, which is covered in a later unit

© Copyright IBM Corporation 2013

Figure 4-41. Log files and locations for HPEL

WA5913.0

Notes:

HPEL log data repository:

The log data repository is a storage facility for log records. Log data is typically intended for review by administrators. This data includes any information that the server or applications write to System.out, System.err, or java.util.logging at level Detail or higher. (It includes Detail, Config, Info, Audit, Warning, Severe, Fatal, and any custom levels at level Detail or higher.)

HPEL trace data repository:

The trace data repository is a storage facility for trace records. Trace data is typically intended for use by application programmers or by the WebSphere Application Server support team. This data includes any information that the server or applications write to java.util.logging at levels below level Detail (including Fine, Finer, Finest, and any custom levels below level Detail).

HPEL text log:

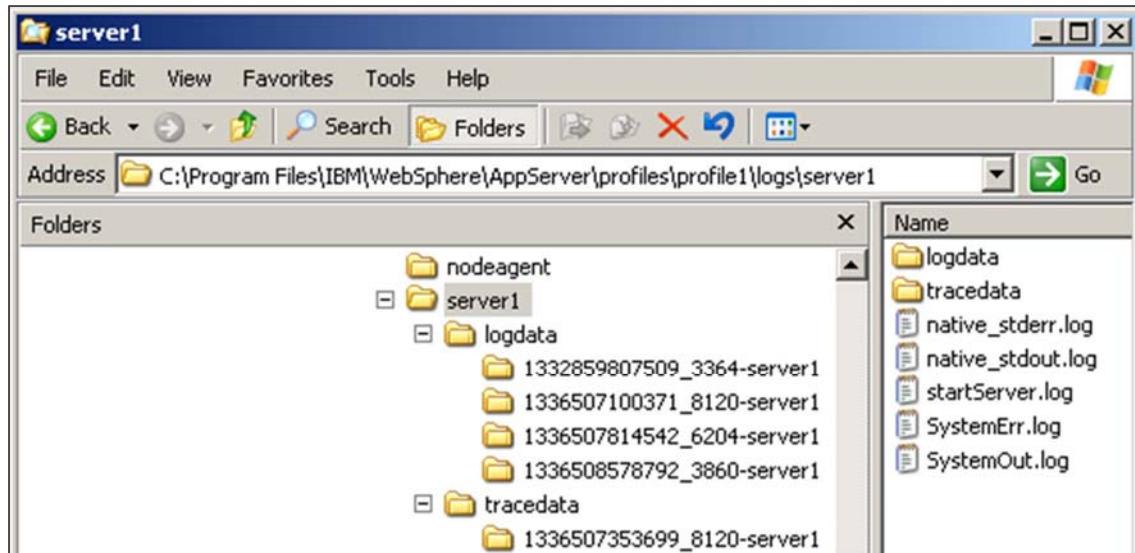
The text log file is a plain text file for log and trace records. The text log file is provided for convenience, primarily so that log content can be read without having to run the LogViewer command-line tool to convert the log data repository content to plain text.

The text log file does not contain any content that is not also stored in either the log data repository or trace data repository. You can disable the text log to enhance server performance. The text log can be configured to record trace content for debugging convenience.



Contents of the logs directory

- The `logdata` directory contains the `standard.out`, `standard.err`, and log messages for each instance of the server
- The `tracedata` directory contains trace data for each instance of the server



© Copyright IBM Corporation 2013

Figure 4-42. Contents of the logs directory

WA5913.0

Notes:

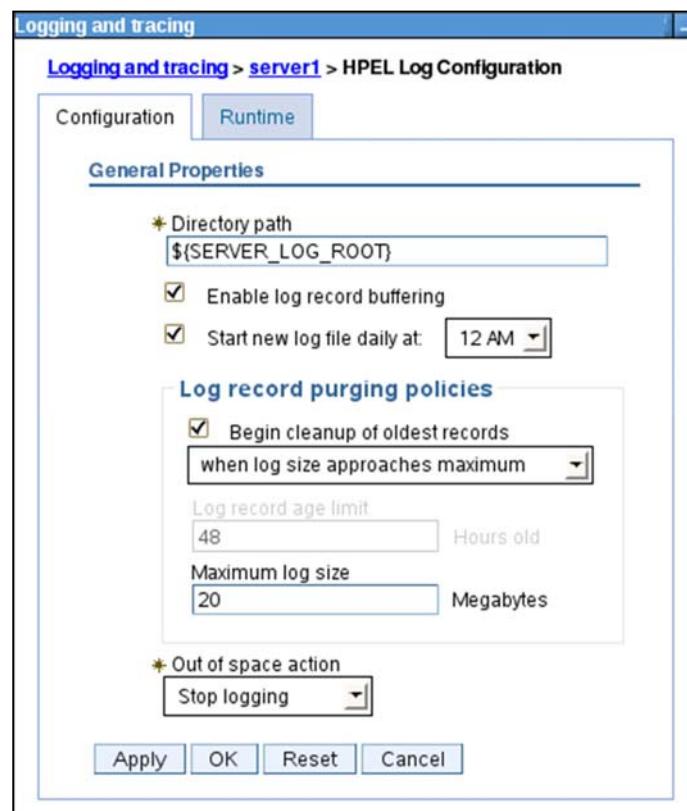
In the `logdata` directory, there is a subdirectory for each instance of the server. The subdirectory name contains the PID of the server instance.

The directory for `server1` contains the `pid` file, `native_stderr`, `native_stdout`, and `start/stopServer` log files, as in previous versions of WebSphere Application Server.



Configure HPEL logging

- Clicking the **Configure HPEL logging** link starts the configuration page
- Changes that are made on the Configuration tab require a server restart
- Changes that are made on the Runtime tab take effect immediately



© Copyright IBM Corporation 2013

Figure 4-43. Configure HPEL logging

WA5913.0

Notes:

Enable log record buffering: Specifies that the logging system avoids writing to disk each time a log record is created. The logging system creates a buffer that can hold a number of log records, and writes the buffered events when the buffer is full. The logging system also writes the buffered events after a few seconds pass, even if the buffer is not full. Selecting this setting significantly improves logging performance; however, if the server stops unexpectedly, the contents might not be written to the log repository.

Begin cleanup of oldest records: Specifies the log cleanup settings to be used to automatically purge the oldest log records, or log records that no longer fit in the configured space, from the log repository.

Out of space action: Specifies how the server reacts to an inability to add content to the log repository.



HPEL Log Viewer

- Click Troubleshooting > Logs and trace > *server_name* > View HPEL logs and trace
 - There are numerous filtering options to modify which records are shown

Logging and tracing

[Logging and tracing](#) > [server1](#) > Log Viewer

Content and Filtering Details

Select record in the log and click to show all records from the same thread

Refresh View Show Only Selected Threads Show All Threads Select Columns ... Export ... Copy

Viewing log records from server instance June 3, 2011 09:55:44

Number of records to show: 20 First Page

TimeStamp	Thread ID	Logger	Level	Message
6/3/11 09:55:44.896	00000000	com.ibm.ejs.ras.ManagerAdmin	INFO	TRAS0017I : The startup trace state is *=info.
6/3/11 09:55:44.940	00000000	com.ibm.ejs.ras.ManagerAdmin	INFO	TRAS0111I : The message IDs that are in use are c
6/3/11 09:55:45.094	00000000	com.ibm.ws.config.ModelMgr	INFO	WSVR0800I : Initializing core configuration models

Full logger names

© Copyright IBM Corporation 2013

Figure 4-44. HPEL Log Viewer

WA5913.0

Notes:

The log view section displays the records. Use the First Page, Previous Page, Next Page, and Last Page buttons to move through the list of records. You can also specify filter criteria in the Content and Filtering Details section to limit the rows that are displayed. Records are always displayed in the order that the server recorded them. By default the log view has five columns:

- Time Stamp:** The time when the event was recorded.
- Thread ID:** The identity of the thread that recorded the event in hexadecimal notation.
- Logger:** The logger that recorded the event.
- Level:** The type of event that was recorded.
- Message:** The message from the recorded event. If the message has a message ID, the message ID is underlined. Click the message ID to get an explanation and suggested user action for the message.

To manipulate the log view, you can use available buttons to complete the following actions:

- **Refresh View:** Clears the contents of the viewer and reinitializes the view by using records from the server. Use this button to retrieve information about any additional rows that are created since the log viewer was started.
- **Show Only Selected Threads:** Filters any records that are created by any thread other than the one selected in the selection area. Clicking this button enables the Show All Threads button.
- **Show All Threads:** Displays any records that were filtered when you clicked Show Only Selected Threads. This button is enabled only when you use the Show Only Selected Threads button to restrict the view.
- **Select Columns:** Enables you to select the columns in the viewer that you want to view.
- **Export:** Exports logs to a local workstation in any of basic, advanced, or binary (HPEL) formats.
- **Copy to Clipboard:** Copies the records that are highlighted in the selection area into the operating system clipboard.
- **Server Instance Information:** Displays attributes for the selected server instance process. Use this table to find attributes and corresponding values for the server instance process environment. These properties are similar to the ones found in the header of basic mode logs.



HPEL Log Viewer: Filtering options

- Server Instance
- View Contents
- Levels
 - Info
 - Audit
 - Config
 - Detail
 - Warning
 - Fatal
 - Severe
 - Fine
 - Finer
 - Finest
- Filtering
- Event Timing

Logging and tracing

Logging and tracing > server1 > Log Viewer

Content and Filtering Details

Server Instance

Server instances grouped by server start date and time:

- March 27, 2012
 - 15:47:53
 - 16:11:13
 - 21:37:29
- April 13, 2012
 - 12:15:59

View Contents

System out
 System err
 Logs and trace

Minimum level:

Maximum level:

Filtering

Wild cards: * , ?, % are allowed
 Separate multiple entries by a ':'
 Include loggers:
 Exclude loggers:
 Message contents:

Event Timing

From: On:
 Until: On:

© Copyright IBM Corporation 2013

Figure 4-45. HPEL Log Viewer: Filtering options

WA5913.0

Notes:

- **Server instance:** A server instance represents a run of a server process. Each time the server is restarted, a new server instance is created. By default, the log view table shows log records that are generated in the most recent server instance. To select a different server start time, choose a server instance with the appropriate start time stamp. The time stamps that are shown represent the time stamp of the first record that is written to each server instance. Select a particular server instance to change the server instance from which log records are retrieved.
- **View Contents:** Controls what content sources are displayed in the log view.
- **System Out:** Specifies that content that is logged to the `System.out` output stream is included in the log view.
- **System Error:** Specifies that content that is logged to the `System.err` output stream is included in the log view.
- **Logs and trace:** Specifies that log and trace records are included in the log view. Log and trace entries can be further specified to include a minimum or maximum level. Minimum and maximum can be specified together, for example to display only a certain level of trace. If you do not select Logs and trace, then no log or trace records of any severity can be displayed.

- **Filtering:** Controls which records are included in and excluded from the log view. For all filters in this section, a colon (:) can be used as a separator character to specify multiple entries. A limited set of regular expression characters can be used. Refer to console documentation for more details. If multiple filter settings are specified, the filter conditions must all be true for a record to be displayed in the log view.
- **Include loggers:** Specifies the list of loggers whose records are included in the log view.
- **Exclude loggers:** Specifies the list of loggers whose records are excluded from the log view.
- **Message contents:** Specifies the message content that each record must contain to be included in the log view.
- **Event timing:** Controls what records are displayed in the log view, as is based on a start and end date and time.

HPEL LogViewer: Command-line tool (1 of 2)

- Use the `LogViewer` command to query the contents of the HPEL log and trace repositories
 - `logViewer.sh/bat`
- You can also use the `LogViewer` command to view new log and trace repository entries as the server writes content to them
 - `logViewer.sh/bat -monitor [interval]`
- There are many command-line options for filtering the log and trace data
 - `-format` Specifies the output format
 - `-thread` Shows log entries from a specific thread
 - `-latestInstance` Gets log and trace data from the recent server instance
 - `-level` Extracts log entries from the specified level
 - `-minLevel` Shows log entries that are below the specified level
 - `-maxLevel` Shows log entries that are above the specified level
 - others

© Copyright IBM Corporation 2013

Figure 4-46. HPEL LogViewer: Command-line tool (1 of 2)

WA5913.0

Notes:

The first tool for analyzing HPEL logs is the command-line log viewer. This tool is simple, intuitive, and fast for doing analysis on the logs in problem determination efforts. The user can be unaware of whether data is written to `SystemOut.log`, `SystemErr.log`, or `trace.log`. With HPEL, the logs are consolidated and the log viewer can be used to view all the data or it can be filtered according to what subset of messages are needed.

- `-monitor [interval]`

Specifies that you want the LogViewer to continuously monitor the repository and output new log record entries as they are created. You can provide an optional integer argument after this parameter to specify how often you want the LogViewer tool to query the repository for new records. By default the LogViewer queries the repository for new records every 5 seconds. When used with other filtering options, only those new records that match the filter criteria are displayed.

- `-level FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`

Specifies that LogViewer must extract log entries only from the specified level. If combined with `-minLevel` or `-maxLevel`, the last option or options are used.

- `-minLevel FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`

Specifies that LogViewer must not show log entries that are below the specified level.

Specifying a level extracts all messages at that level and the ones above it.

- `-maxLevel FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`

Specifies that LogViewer must not show log entries that are above the specified level.

Specifying a level extracts all messages at that level and the ones below it.

- `-format <basic | advanced | CBE-1.0.1>`

Specifies the output format. Supported formats include basic, advanced, and the `CBE-1.0.1` format. If you do not include this parameter, the output is in basic format.

- `-thread <thread_id>`

Displays log entries from a specific thread. This option filters out any log messages that were not created according to the thread ID that you specified. Note: Specify the thread ID in hexadecimal format.

- `-latestInstance`

Retrieves the log and trace data from the most recent server instance.

HPEL LogViewer: Command-line tool (2 of 2)

- To view log and trace data in ADVANCED format for the latest server instance and a specific thread ID

— logViewer.sh -lastestInstance -format Advanced -Thread 88

```
[root@washost bin]# ./logViewer.sh -latestInstance -format Advanced -Thread 88
```

Using /opt/IBM/WebSphere/AppServer/profiles/profile1/logs/server1 as repository directory.

```
***** Start Display Current Environment *****
. < Environment data removed>
```

```
***** End Display Current Environment *****
```

```
[8/1/13 12:49:25.878 EDT] 00000088 I UOW= source=com.ibm.ws.wsgroup.p2p.P2P
Group org=IBM prod=WebSphere component=Application Server thread=[Thread-63]
ODCF8040I: Detected process washostCell\washostNode01\nodeagent started.
```

```
[8/1/13 12:49:25.879 EDT] 00000088 I UOW= source=com.ibm.ws.wsgroup.p2p.P2P
Group org=IBM prod=WebSphere component=Application Server thread=[Thread-63]
ODCF8040I: Detected process washostCell\washostCellManager\dmgr started.
```

Operation Complete

Processed 2 records in 0.052 seconds (38.462 records per second).

```
[root@washost bin]#
```

© Copyright IBM Corporation 2013

Figure 4-47. HPEL LogViewer: Command-line tool (2 of 2)

WA5913.0

Notes:

With the advanced format, complete logger names can be seen as UOW=source.

What is Cross Component Trace (XCT)?

- XCT is a feature that annotates the logs so that entries that are related to a request are identified as belonging to the same unit of work
- The request might traverse more than one:
 - Thread
 - Process
 - Server
- XCT helps identify the root cause of problems across components, which provides the following benefits:
 - Enables administrators and support teams to follow the flow of a request from end-to-end as it traverses thread or process boundaries, or travels between stack products and WebSphere Application Server
 - Helps to resolve questions about which component is responsible for a request that fails

© Copyright IBM Corporation 2013

Figure 4-48. What is Cross Component Trace (XCT)?

WA5913.0

Notes:

Depending on the nature of your applications, multiple threads within an application server can be used to handle requests, such as HTTP requests or JMS requests. More than one application server might handle some requests, such as when one application server makes a request to another application server for a web services request.

Applications that are built by using distributed architectures, such as service-oriented architecture, can benefit from XCT, since XCT helps facilitate problem determination across multiple services on different systems.

Administering XCT

- XCT can be enabled for a server that uses HPEL or Basic mode
- Click **Troubleshooting > Logs and trace > *server_name* > Change log detail levels**
- Check **Enable log and trace correlation**

Correlation

Enable log and trace correlation so entries that are serviced by more than one thread, process, or server will be identified as belonging to the same unit of work.

Enable log and trace correlation

Include request IDs in log and trace records

Include request IDs in log and trace records and create correlation log records

Include request IDs in log and trace records, create correlation log records, and capture data snapshots

- Select option for including request IDs, creating correlation logs, and capturing data snapshots

© Copyright IBM Corporation 2013

Figure 4-49. Administering XCT

WA5913.0

Notes:

Enable XCT to include request IDs in log and trace files when you want to see which log and trace entries, in all threads and application server processes, are related to the same request. Request IDs are recorded only when using HPEL log and trace mode, and they can be seen or used for filtering by using the `logViewer` command.

Enable XCT to create correlation log records when you want to log how requests branch between threads and processes, and see extra information about each request. Enabling XCT to create correlation log records might have a significant performance impact on your system, so is best suited to test and development environments.

Enable XCT to capture data snapshots when you want to store entire request and response bodies to the file system. Enabling XCT to capture data snapshots might have a significant performance impact on your system, so is best suited to test and development environments. XCT captures data snapshots for message requests and responses that the SIBus handles.

XCT request IDs

- XCT request IDs are identifiers added to log and trace records that the server produces
- XCT adds the same request ID to every log or trace record while the record is a part of the same request, regardless of which thread or JVM produces the log or trace entry
- When XCT is used with the HPEL log and trace infrastructure, you can view request IDs with the logViewer tool when logs are output in advanced format
 - `logViewer.sh -minLevel WARNING -format advanced`

```
[Time_stamp] 00000094 W UOW= source=com.ibm.ws.webcontainer.srt  
class=com.ibm.ws.webcontainer.srt.SRTServletResponse method=setIntHeader  
org= prod= component= thread=[WebContainer : 4]  
requestID=[AAAsirk1Njr-AAAAAAA+] appName=[PlantsByWebSphere]
```

© Copyright IBM Corporation 2013

Figure 4-50. XCT request IDs

WA5913.0

Notes:

You can use XCT to augment your log and trace files with correlation information. This correlation information clarifies which threads and which application server processes participated in the handling of each request.

Use XCT request ID information to track requests

- Filter your logs by request ID by using the HPEL LogViewer command-line tool
 - `logViewer.sh -includeExtensions requestId=AAAsirk1Njr-AAAAAAA+AAA+`

```
[Time_stamp] 00000094 XCT           I   BEGIN AAAsirk1Njr-AAAAAAA+AAA+
00000000000-cccccccccc2 HTTPCF(InboundRequest
/PlantsByWebSphere/javax.faces.resource/jsf.js.jsf
RemoteAddress(127.0.0.1) RequestContext(-957274864))

[Time_stamp] 00000094 srt           W
com.ibm.ws.webcontainer.srt.SRTServletResponse setIntHeader SRVE8094W:
WARNING: Cannot set header. Response already committed.

[Time_stamp] 00000094 XCT           I   END   AAAsirk1Njr-AAAAAAA+AAA+
00000000000-cccccccccc2 HTTPCF(Request AsyncWrite RequestContext(-
957274864))
```

© Copyright IBM Corporation 2013

Figure 4-51. Use XCT request ID information to track requests

WA5913.0

Notes:

XCT log records:

XCT log records are typically added to the logs to:

- Demarcate the beginning and ending of work for a particular request on a particular thread
- Demarcate when work is about to transfer to another thread or process, or to indicate when work returned from another thread or process
- Demarcate when work moves from major component to major component, even if work continues on the same thread; for example to show transfer of control from application server code to application code

XCT log records are composed of:

- XCT type (BEGIN / END)
- XCT parent correlator ID (for example, 00000000000-cccccccccc2)
- XCT current correlator ID (for example, AAAsirk1Njr-AAAAAAA+AAA+)

- XCT annotations (for example, `HTTPCF(InboundRequest/PlantsByWebSphere/
javax.faces.resource/jsf.js.jsf RemoteAddress(127.0.0.1)
RequestContext(-957274864))`)

XCT tools:

The HPEL logViewer tool is able to filter log and trace records by request ID.

Tools, such as the XCT Log Viewer, can also take advantage of XCT log records or XCT request IDs, or both, when rendering log and trace content. The XCT Log Viewer is available as a tool add-on for the IBM Support Assistant.

4.5. Gathering JVM-related data

Gathering JVM-related data



© Copyright IBM Corporation 2013

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

8.0

Figure 4-52. Gathering JVM-related data

WA5913.0

Notes:

Enable verbose garbage collection

- The JVM runtime provides verbose GC as an option
- Enables a garbage collection log that provides data such as:
 - Interval between collections
 - Duration of collection
 - Whether compaction was required
 - Memory size, memory that is freed, memory available
- Enable the verbose GC by using the administrative console
 - Servers > server_name > Process Definition > Java Virtual Machine**
 - Select **Verbose Garbage Collection** check box on configuration or runtime tab



© Copyright IBM Corporation 2013

Figure 4-53. Enable verbose garbage collection

WA5913.0

Notes:

It is often suggested that you have verbose GC enabled permanently in production. The performance cost on a reasonably well-tuned JVM is 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 as a way to monitor the health of the system, even when nothing bad is noticed.

Enabling verbose GC is a decision that each system administrator must make conscientiously. But it is no longer plainly “not recommended as a normal production setting.”

It usually writes to native_stderr.log file. It varies depending on platform and WebSphere version. There is some processor cost because of disk I/O, but usually minimal unless thrashing.



Java dumps and cores

- New feature in the Troubleshooting section is **Java dumps and cores**

The screenshot shows the 'Troubleshooting' section of the WebSphere Application Server interface. On the left, under 'Logs and trace', the 'Java dumps and cores' link is highlighted with a red box and an arrow pointing to the main panel. The main panel has three tabs at the top: 'Heap dump', 'Java core', and 'System dump', also highlighted with a red box. Below the tabs are four icons: a file with a checkmark, a file with a minus sign, a downward arrow, and an upward arrow. Underneath are dropdown menus for 'Select' (Server) and 'Node'. A message says 'You can administer the following resources:' followed by a table:

<input type="checkbox"/>	dmgr	was8hostCellManager01
<input type="checkbox"/>	myserver1	was8hostNode01
<input type="checkbox"/>	nodeagent	was8hostNode01
<input type="checkbox"/>	nodeagent	was8hostNode02
<input type="checkbox"/>	server1	was8hostNode01
Total 5		

© Copyright IBM Corporation 2013

Figure 4-54. Java dumps and cores

WA5913.0

Notes:

Clicking the Java dumps and cores link starts the panel that is shown in the screen capture. Use this panel to generate heap dumps, javacores, or system dumps for a running process. Select the server and click the appropriate button for Heap dump, javacore (thread dump), or System dump (JVM core). The files that result from these operations are placed on the local file system, by default they are written to the profile root directory, `was_root/profiles/profile-name`.

- Heap dump:** A heap dump is a snapshot of JVM memory. It shows live objects in the memory and references between them.
- Javacore:** Use this button to investigate why a server is hanging or investigate messages in the logs that indicate a thread did not complete its work in the expected amount of time.
- System dump:** Use this button to generate system native dumps of the server process. These memory dumps can be large.
- A note on verbose garbage collection data:** As in previous versions, verbose GC is not enabled by default. When you enable verbose GC for a server in V8, the default garbage collection policy is generational-concurrent (gencon). The data is written to the `native_stderr.log` or `native_stdout.log` depending on the operating system of the server.

Dumping the JNDI name space

- dumpNameSpace utility shows JNDI directory content
- Useful to ensure correct association of named objects:
 - JDBC resources
 - EJBs
 - JMS resources
 - Other resources
- Syntax and some of the options:

```
<was_root>/bin/dumpNameSpace
  [-host bootstrap_host_name (defaults to localhost)]
  [-port bootstrap_port_number (defaults to 2809)]
  [-startAt subcontext/in/the/tree]
```

- Output can be redirected to a file and inspected

© Copyright IBM Corporation 2013

Figure 4-55. Dumping the JNDI name space

WA5913.0

Notes:

- Usage: dumpNameSpace [-keyword value]
 - If a keyword occurs more than once with different values, the value that is associated with the last occurrence is used.
 - The keywords and associated values are: -host myhost.austin.ibm.com
 - Bootstrap host is the WebSphere host whose namespace you want to dump. It defaults to “localhost.”
 - port nnn
 - Bootstrap port, defaults to 2809.
 - factory com.ibm.websphere.naming.WsnInitialContextFactory
 - The initial context factory to be used to get the JNDI initial context. It defaults as shown and normally does not need to be changed.
 - root [cell | server | node | host | legacy | tree | default]

Dumping the JNDI name space example

- Enter command: `dumpNameSpace -root server -port 9810`

```
Name Space Dump
Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
Provider URL: corbaloc:iiop:localhost:9810
Requested root context: server
Starting context: (top)=was8host01Cell01/nodes/was8host01Node01/servers/server1
Formatting rules: jndi
Time of dump: Fri Apr 13 12:27:55 EDT 2012
=====
Beginning of Name Space Dump
=====
1 (top)                                     javax.naming.Context
2 (top)/com                                 javax.naming.Context
3 (top)/com/ibm                             javax.naming.Context
4 (top)/com/ibm/websphere                     javax.naming.Context
...
28 (top)/jdbc                                javax.naming.Context
29 (top)/jdbc/lsched                          javax.resource.cci.ConnectionFactory
30 (top)/jdbc/pgc                            javax.resource.cci.ConnectionFactory
31 (top)/jdbc/DefaultEJBTimerDataSource      javax.resource.cci.ConnectionFactory
32 (top)/jdbc/PlantsByWebSphereDataSource    javax.resource.cci.ConnectionFactory
...
41 (top)/jta                                  javax.naming.Context
42 (top)/jta/usertransaction                  java.lang.Object
43 (top)/ejb                                  javax.naming.Context
44 (top)/ejb/ivtEJBObject                   com.ibm.websphere.ivt.ivtEJB.ivtEJBHome
45 (top)/eis                                  javax.naming.Context
46 (top)/eis/DefaultDatasource_CMP           javax.resource.cci.ConnectionFactory
```

© Copyright IBM Corporation 2013

Figure 4-56. Dumping the JNDI name space example

WA5913.0

Notes:

The figure shows an example of the JNDI namespace dump. Note: the entire memory dump is not shown on the slide because of space limitations.

Unit summary

Having completed this unit, you should be able to:

- Identify and access several resources for problem investigation
- Identify, locate, and configure server log files
- Enable HPEL logging for a server and use log viewer tools
- Describe and use cross-component trace (XCT)
- Generate JVM-related diagnostic data by using the administrative console and other tools

© Copyright IBM Corporation 2013

Figure 4-57. Unit summary

WA5913.0

Notes:



Checkpoint questions

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

© Copyright IBM Corporation 2013

Figure 4-58. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

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

Checkpoint answers

1. The following resources are available:
 - IBM Support Assistant
 - IBM Support Portal
 - Information centers
 - More
2. A MustGather document tells you the specific information that you must collect for a particular type of problem.
3. `<was_root>/profiles/<profile_name>/logs`
4. Trace output can be directed to a file or a ring buffer.
5. False: Tracing *can* be started while the server is running.

© Copyright IBM Corporation 2013

Figure 4-59. Checkpoint answers

WA5913.0

Notes:

Exercise 2

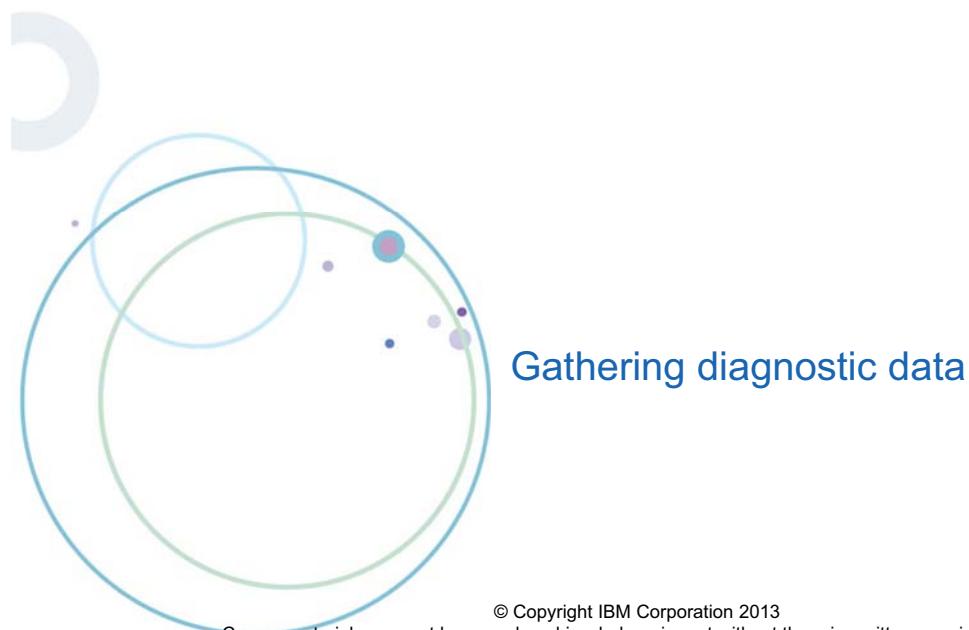


Figure 4-60. Exercise 2

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Configure basic (legacy) logging and examine server log files
- Enable HPEL logging for an application server
- Enable tracing on an application server component and read a trace output log
- Examine logs and trace data by using the HPEL Log Viewer
- Enable and configure Cross Component Trace (XCT)
- Use the administrative console troubleshooting section to view runtime messages and configuration problems
- Enable and use diagnostic providers in the administrative console
- Examine FFDC logs
- Use the Classloader viewer in the administrative console
- Compare namespaces from different contexts by using the dumpNameSpace tool and the name server MBean

© Copyright IBM Corporation 2013

Figure 4-61. Exercise objectives

WA5913.0

Notes:

Unit 5. Introduction to JVM-related problems

What this unit is about

This unit describes the functions of a JVM and provides an overview of the common types of JVM-related problems.

What you should be able to do

After completing this unit, you should be able to:

- Describe components of JVM and overall architecture (Java Development Kit V6.0)
- Describe garbage collection (GC) and GC tuning policies
- Describe JVM command-line arguments
- Describe javacore files and how to obtain them
- Identify a sluggish JVM and detect bottleneck problems
- Explain how to tune the heap size
- Use JVM-related tools: Garbage Collection and Memory Visualizer (GCMV), Memory Analyzer tool (MAT), and Java Health Center

How you will check your progress

- Checkpoint questions
- Lab exercise

References

IBM Redbooks

WebSphere Application Server V6.1 Problem Determination: IBM Redpaper Collection, SG24-7461-00:
<http://www.redbooks.ibm.com/abstracts/sg247461.html?Open>

WebSphere Application Server V7 Administration and Configuration Guide, SG24-7615-00:
<http://www.redbooks.ibm.com/abstracts/sg247615.html?Open>

IBM JVM Diagnosis documentation:

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

IBM JRE and JDK forum:

[http://www.ibm.com/developerworks/forums/
forum.jspa?forumID=367](http://www.ibm.com/developerworks/forums/forum.jspa?forumID=367)

IBM Monitoring and Diagnostic Tools for Java – Health Center:

[http://www.ibm.com/developerworks/java/jdk/
tools/healthcenter/](http://www.ibm.com/developerworks/java/jdk/tools/healthcenter/)

Garbage collection policy and tuning article on developerWorks:

[http://www.ibm.com/developerworks/java/
library/j-ibmjava2/index.html](http://www.ibm.com/developerworks/java/library/j-ibmjava2/index.html)

IBM Guided Activity Assistant:

IBM Guided Activity Assistant contains much content for
JVM problem determination:

[http://www.ibm.com/support/
docview.wss?uid=swg24011818](http://www.ibm.com/support/docview.wss?uid=swg24011818)

IBM WebSphere Application Server V6.1 on the Solaris 10 Operating
System, Redbooks, SG24-7584-00

Unit objectives

After completing this unit, you should be able to:

- Describe components of JVM and overall architecture (Java Development Kit V6.0)
- Describe garbage collection (GC) and GC tuning policies
- Describe JVM command-line arguments
- Describe javacore files and how to obtain them
- Identify a sluggish JVM and detect bottleneck problems
- Explain how to tune the heap size
- Use JVM-related tools: Garbage Collection and Memory Visualizer (GCMV), Memory Analyzer tool (MAT), and Java Health Center

© Copyright IBM Corporation 2013

Figure 5-1. Unit objectives

WA5913.0

Notes:



Topics

- JVM introduction
- Introduction to JVM problem determination
- JVM tuning

© Copyright IBM Corporation 2013

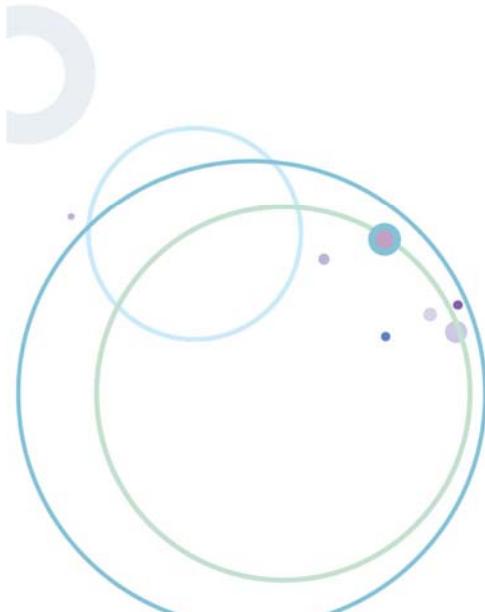
Figure 5-2. Topics

WA5913.0

Notes:

5.1. JVM introduction

JVM introduction



© Copyright IBM Corporation 2013

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

8.0

Figure 5-3. JVM introduction

WA5913.0

Notes:

Java virtual machine (JVM) features

Almost every WebSphere process runs in a JVM

The JVM provides:

- Class loading
 - A class loader verifies and loads classes into memory
 - Multiple class loaders are involved in loading the required libraries for an application to run
 - Each class loader loads its own classes
- Garbage collection (GC)
 - Garbage collection takes care of memory management for the entire application server
 - The GC process searches memory to reclaim space from program segments or inactive Java object
- Execution management
 - Manages the bookkeeping work for all the Java threads
- Execution engine (Interpreter)
 - Interprets the Java methods

© Copyright IBM Corporation 2013

Figure 5-4. Java virtual machine (JVM) features

WA5913.0

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 bytecode into the native instructions of the host computer. The application server, being a Java process, requires a JVM 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 an interface that is independent of the underlying operating system and computer 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 bytecode that targets the abstract virtual machine; the JVM is responsible for running the bytecode on the specific operating system and hardware combinations.

Just-in-time compiler (JIT) basics

- The just-in-time compiler (JIT) is essential for a high-performing Java application
 - Java is write-once-run-anywhere; thus it is interpretive by nature and without the JIT, cannot compete with native code applications
- The JIT works by compiling bytecode that is loaded from the class loader when an application accesses it
 - Because different operating systems have different JIT compilers, there is no standard procedure for when a method is compiled
 - As your code accesses methods, the JIT determines how frequently specific methods are accessed
 - Methods that are used often are compiled to optimize performance

© Copyright IBM Corporation 2013

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

WA5913.0

Notes:

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

Ahead-Of-Time (AOT) compiler basics

- Ahead-Of-Time (AOT) compiles Java classes into native code for subsequent executions of the same program
 - The AOT compiler works with the class data sharing framework
- The AOT compiler generates native code dynamically while an application runs and caches any generated AOT code in the shared data cache
 - Subsequent JVMs that run the method, can load and use the AOT code from the shared data cache without incurring the performance decrease experienced with JIT-compiled native code
- The AOT compiler is enabled by default, but is only active when shared classes are enabled
 - By default, shared classes are disabled so that no AOT activity occurs
 - The `-Xshareclasses` command-line option can be used to enable shared classes
- When the AOT compiler is active, the compiler selects the methods to be AOT compiled with the primary goal of improving startup time

© Copyright IBM Corporation 2013

Figure 5-6. Ahead-Of-Time (AOT) compiler basics

WA5913.0

Notes:

Because AOT code must persist over different program executions, AOT-generated code does run as fast as JIT-generated code. AOT code usually runs faster than interpreted code.

In a JVM without an AOT compiler or with the AOT compiler disabled, the JIT compiler selectively compiles frequently used methods into optimized native code.

A time cost is associated with compiling methods because the JIT compiler operates while the application is running. Because methods begin by being interpreted and most JIT compilations occur during startup, startup times can be increased.

Startup performance can be improved by using the shared AOT code to provide native code without compiling. There is a small time cost to load the AOT code for a method from the shared data cache and bind it into a running program. The time cost is low compared to the time it takes the JIT compiler to compile that method.

The `-Xshareclasses` option can be used to enable shared classes, which might also activate the AOT compiler if AOT is enabled.

JVM version (1 of 2)

- WebSphere supports several JVMs based on version and operating system type
 - Windows, AIX, and Linux: IBM supplied
 - Can use only IBM SDK that zWSAS provides on z/OS
 - Solaris and HP-UX: Hybrid of IBM add-ons and vendor supplied JVM
- For a comprehensive list of the supported JVMs, check
 - <http://www.ibm.com/support/docview.wss?uid=swg27038218>
- 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`
- JVM version can be found in the server job logon z/OS

© Copyright IBM Corporation 2013

Figure 5-7. JVM version (1 of 2)

WA5913.0

Notes:

JVMs differ on the available tools that are supported and can have different behavior, even between different versions that are 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 V7.0 supports the Java Development Kit (JDK) Version 6.0.

WebSphere Application Server V6.1 supports the Java Development Kit (JDK) Version 5.0. This version can be different from JDK version 1.4.2, which is WebSphere Application Server V6.0 and earlier supports. 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 identifies the supported JDKs. The WebSphere supplied Java 2 SDK is required for both the runtime 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-UX and Solaris 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 Oracle Solaris, WebSphere Application Server contains an embedded copy of the Oracle 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 Oracle and IBM products. However, the core JVM and JIT are Oracle 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 JVMs, but only when it is an embedded part of IBM middleware (for example, WebSphere Application Server).

JVM version (2 of 2)

- To determine the JVM version in use:
 - Run `java -fullversion` (IBM JVMs only) from the command line

```
[root@washost bin]# ./java -fullversion
java full version "JRE 1.6.0 IBM Linux build pxa6460_26sr5fp1ifix-
20130408_02 (SR5 FP1)"
```

- Run `java -version` from the command line

```
[root@washost bin]# ./java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build pxa6460_26sr5fp1ifix-
20130408_02(SR5 FP1+IV38399+IV38578))
IBM J9 VM (build 2.6, JRE 1.6.0 Linux amd64-64 Compressed References
20130301_140166 (JIT enabled, AOT enabled)
J9VM - R26_Java626_SR5_FP1_20130301_0937_B140166
JIT   - r11.b03_20130131_32403
GC    - R26_Java626_SR5_FP1_20130301_0937_B140166_CMPRSS
J9CL - 20130301_140166)
JCL   - 20130408_01
```

© Copyright IBM Corporation 2013

Figure 5-8. JVM version (2 of 2)

WA5913.0

Notes:

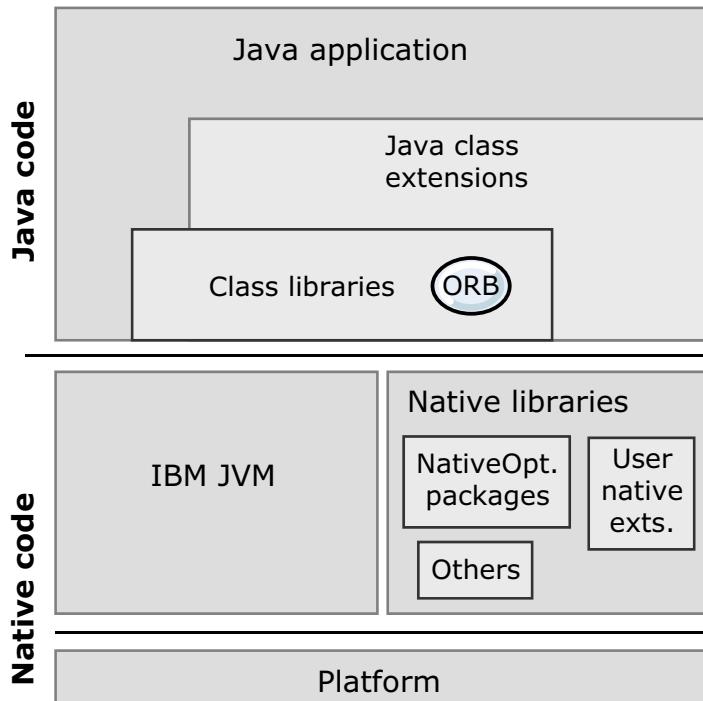
The Java virtual machine (JVM) used by the IBM SDK for Java is the IBM J9 virtual machine (J9 VM).

- JIT is just-in-time compiler.
- GC is garbage collector.
- J9CL is IBM J9 class library.
- JCL is Java Class Library.

JVM overview

- JVM is built by using object-oriented design
- The core run times of JVMs are developed in C and run most functions in native code
 - Garbage collector and memory management
 - JIT
 - I/O subroutines, OS calls
- The J2SE and Java EE APIs all exist at the Java code layer
 - Makes data structures available
 - Gives users access to needed function
 - Allows interactions with system

Java application stack



© Copyright IBM Corporation 2013

Figure 5-9. JVM overview

WA5913.0

Notes:

A Java application uses the Java class libraries that the JRE provides to implement the application-specific logic. The class libraries, in turn, are implemented in terms of other class libraries. Eventually the JVM provides primitive native operations. In addition, some applications must access native code directly.

The diagram on this slide shows the components of a typical Java Application Stack and the IBM JRE.

The JVM facilitates the invocation of native functions by Java applications and a number of well-defined Java Native Interface (JNI) functions for manipulating Java from native code.

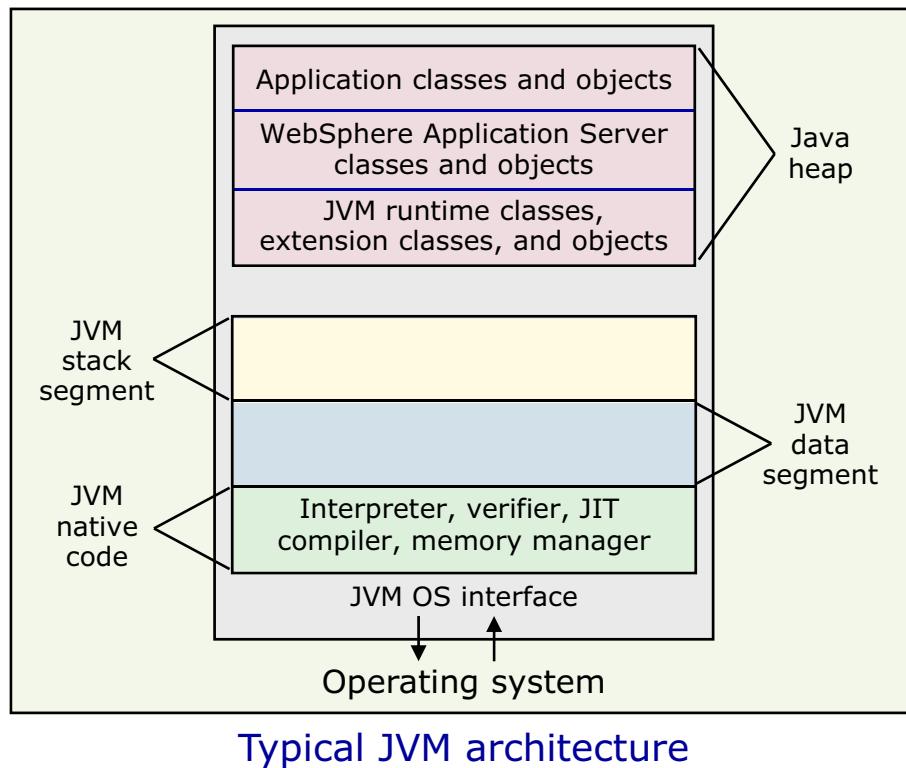
Keep in mind that WebSphere Application Server is a Java application.

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

- Due to different platforms that have 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; to optimize performance, it quickly compiles methods that are touched most frequently.

JVM memory segments



Typical JVM architecture

© Copyright IBM Corporation 2013

Figure 5-10. JVM memory segments

WA5913.0

Notes:

The important points to remember on this slide:

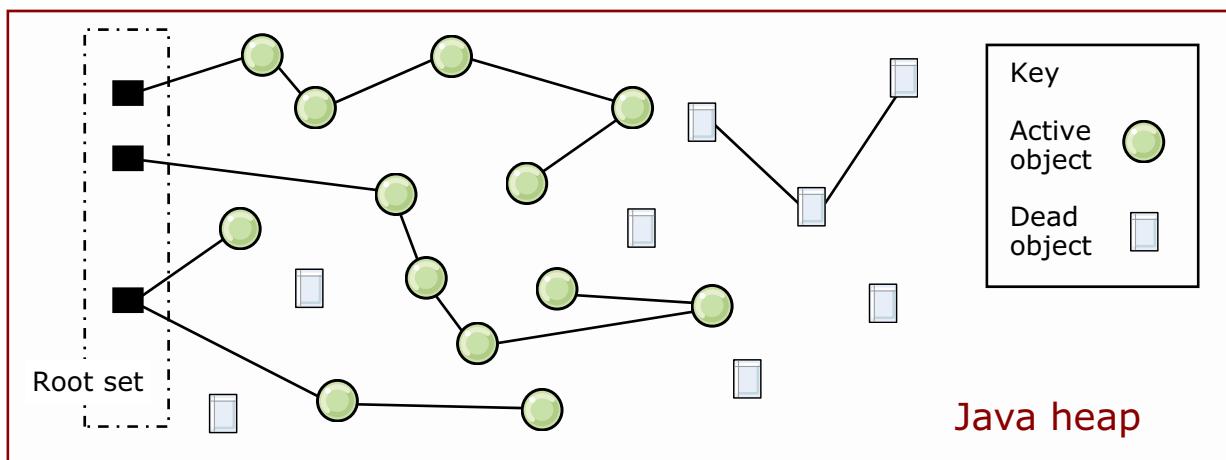
1. The JVM is a process whose memory layout depends on the OS platform. The diagram that is shown is a typical abstraction of process memory layout.
2. As a process, the JVM itself needs its own heap (JVM native heap) from which it gets its own dynamic memory needs.
3. A portion of that heap is allocated for the Java heap where classes and objects are stored. This memory 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 is enough available memory in the JVM heap, the Java heap can be extended easily without acquiring memory from the OS. Otherwise, a request for more memory from the OS is made and the JVM process size increases as necessary.

The data segment contains many native memory buffers used for the network I/O facility and a database buffer if using a type 2 JDBC driver.

Memory that is not allocated to the Java heap is available to the native heap:
Available process memory space minus the Java heap memory equals the native heap.

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



© Copyright IBM Corporation 2013

Figure 5-11. Understanding garbage collection (GC)

WA5913.0

Notes:

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

The GC scans the thread stacks and looks for pointers that appear to be 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 more.

All the objects that these pointers reference are reachable objects; they are listed in a mark vector, which is used to compare with the allocbits vector. The allocbits vector contains the information the GC needs for all of the objects created. From the difference between the mark vector and the allocbits vector, the GC can determine what needs to be collected.

IBM JDK GC process

- **Mark:** Recursively marks all the live objects, starting with the registers and thread stacks
- **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
- **Parallel** mark and sweep process uses main and multiple helper threads (number of processors minus one) to process tasks
- **Concurrent** mark and sweep can be configured
 - It starts a concurrent marking and sweeping phase before the heap is full
 - The mark and sweep phase runs while the application is still running

© Copyright IBM Corporation 2013

Figure 5-12. IBM JDK GC process

WA5913.0

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 that can occur. Because the mark stack has a fixed size, it can overflow. This overflow has a negative impact on pause time:

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

A parallel version of garbage collector mark exists. 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, 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.

Garbage collection policies

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

- **Generational concurrent:** (new default) divides heap into “nursery” and “tenured” segments that provide fast collection for short lived objects
 - Can provide maximum throughput with minimal pause times
- **Optimize for throughput:** Flat heap collector that is focused on maximum throughput
- **Optimize for pause time:** Flat heap collector with concurrent mark and sweep to minimize GC pause time
- **Balanced:** New policy
 - Uses a region-based layout for the Java heap
 - These regions are individually managed to reduce the maximum pause time on large heaps and increase the efficiency of garbage collection
- **Subpool:** This option is now deprecated and is treated as an alias for optimize for throughput

© Copyright IBM Corporation 2013

Figure 5-13. Garbage collection policies

WA5913.0

Notes:

The default policy is “optimize for throughput”. It is a “stop-the-world” policy where the JVM does not do any other work. To optimize the throughput, the garbage collection does all of its work during the GC cycle and does not affect the application when not active.

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

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

The `-Xgcpolicy` options have these effects:

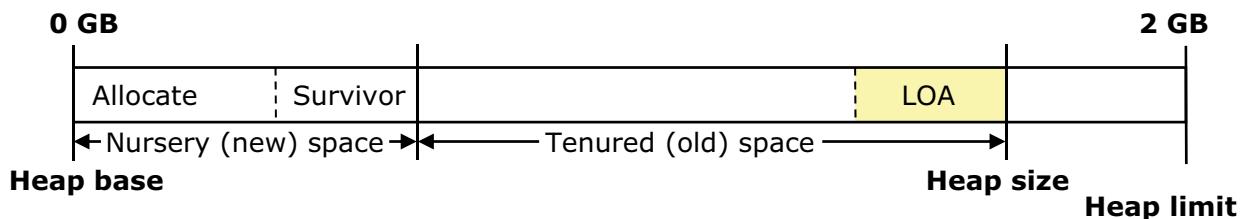
- **Optthrput** disables concurrent mark. If you do not have pause time problems (as seen by erratic application response times), you get the best throughput with this option. *Optthrput* is the default setting.
- **Optavgpause** enables concurrent mark with its default values. If you are having problems with erratic application response times that normal garbage collections cause, 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.

- **Balanced** is a new policy that uses a region-based layout for the Java heap. These regions are individually managed to reduce the maximum pause time on large heaps and increase the efficiency of garbage collection. The policy also uses a different object allocation strategy that improves application throughput on large systems that have Non-Uniform Memory Architecture (NUMA) characteristics
- **Subpool** is used before version 8. 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.

This option is now deprecated. The subpool option is treated as an alias for optimize for throughput. Therefore, if you use this option, the effect is the same as optimize for throughput.

Generational concurrent GC (gencon)

- Similar in concept to the mode that Solaris and HP-UX uses
 - Parallel copy and concurrent global collects by default
- Motivation: Objects die young so focus collection efforts on recently created objects
 - Divide the heap up into a two areas: “new” and “old”
 - Perform allocations from the new area
 - Collections focus on the new area
 - Objects that survive a number of collects in new area are promoted to old area (tenured)



- Ideal for transactional and high data throughput workloads

© Copyright IBM Corporation 2013

Figure 5-14. Generational concurrent GC (gencon)

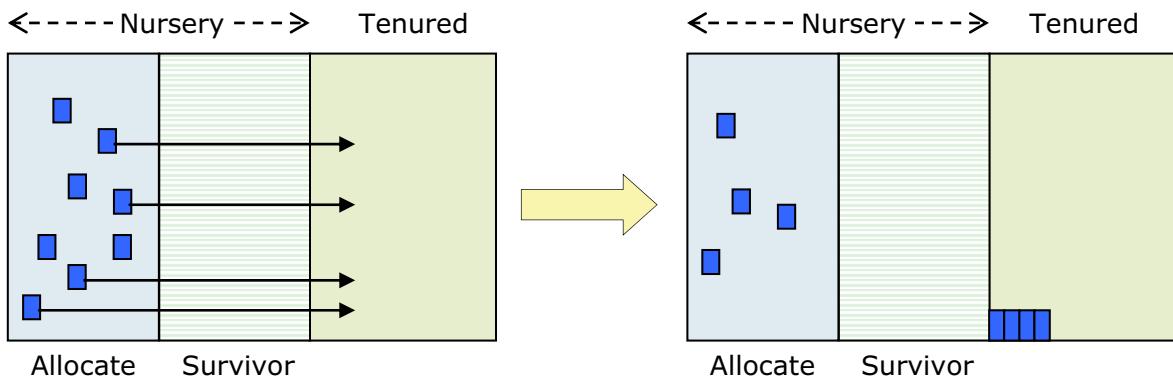
WA5913.0

Notes:

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

Generational concurrent GC: Scavenge (1 of 2)

- When the allocate space is full, a GC is triggered
- The allocate space is traced
- Objects that reach the tenured age (survived a specific number of scavenge operations; maximum is 14) are copied into the tenured area



© Copyright IBM Corporation 2013

Figure 5-15. Generational concurrent GC: Scavenge (1 of 2)

WA5913.0

Notes:

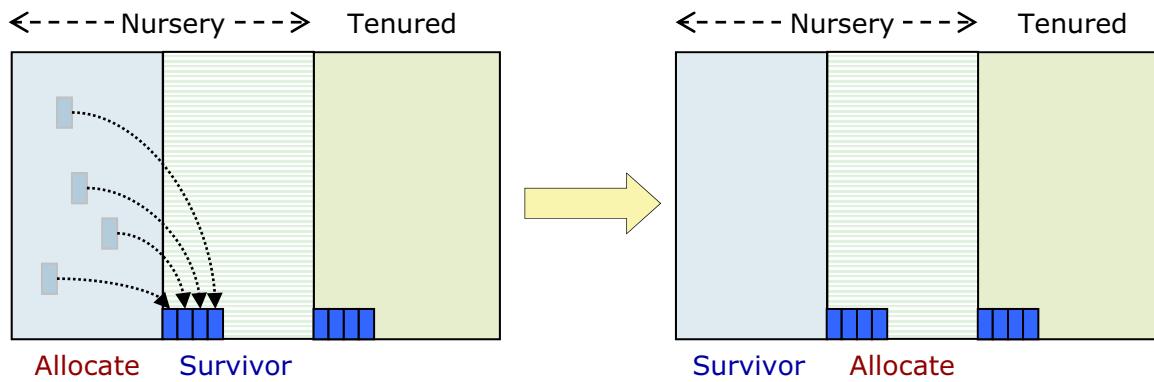
Different algorithms are used in each generation. For young generations, compaction or by-copy algorithms are used. For older tenured generations, the normal mark and sweep algorithms are used.

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. The JVM dynamically adjusts this age, and reaches a maximum value of 14. The age of an object is incremented on each scavenging. A tenure age of x implies that, if the object survived x scavenges between survivor and allocate space, it is promoted. The threshold is adaptive and adjusts the tenure age that is based on the percentage of space that is used in the new area.

Generational concurrent GC: Scavenge (2 of 2)

- Remaining live objects are copied from the allocate space into the survivor space, and a count of the number of times each object was scavenged is incremented
- The copied objects are placed contiguously in the survivor space so that an effective compaction occurs
- At the end of scavenge, the survivor space and allocate space pointers are flipped
 - The survivor space becomes the new allocate space and is empty



© Copyright IBM Corporation 2013

Figure 5-16. Generational concurrent GC: Scavenge (2 of 2)

WA5913.0

Notes:

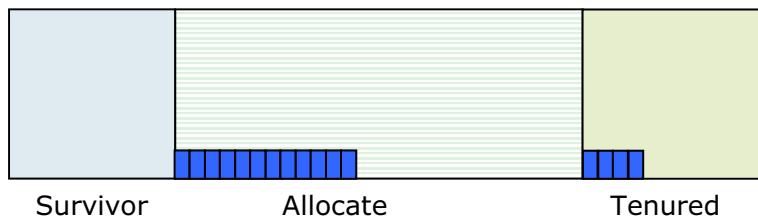
The new area is split into two logical spaces: allocate and survivor. Objects are allocated into the allocate space. When that space is filled, a GC process that is called scavenge is triggered. During a scavenge, live objects are copied either into the survivor space or into the tenured space if they are old enough to reach 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 are 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 or tenured space are now in the “new” allocate space.

A technique that is called *tilting* maximizes the size of the allocate space in the new area. 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.

Generational concurrent GC: Self-adjusting

- The scavenger automatically adjusts the relative sizes of the allocate and survivor spaces by using history and predictions (Tilt)



- The generational collector is designed to respond dynamically to varying conditions so that its behavior is adaptive
- These mechanisms are:
 - Concurrent collection runs in the tenured area
 - In many cases, one rarely or never sees a full classic collection (mark, sweep, compact) in the tenured area
 - The sizes of the young and tenured generations will self-adjust over time based on memory pressure
 - The flip count (number of scavenges that an object must survive before tenure) is adjusted by the scavenger that is based on history
 - The tilt ratio self-adjusts based on history

© Copyright IBM Corporation 2013

Figure 5-17. Generational concurrent GC: Self-adjusting

WA5913.0

Notes:

A technique that is called *tilting* maximizes the size of the allocate space in the new area. 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.

Using IBM Java 7

- In WebSphere Application Server V8.5 and V8.5.5, IBM Java 6 is the default Java runtime
 - You have the option of installing and switching to IBM Java 7
- A new command line tool that is called `managesdk` is added to the `<profile_root>/bin` folder
- List installed run times:
 - `managesdk.sh -listAvailable`
- List run times that are configured for all profiles:
 - `managesdk.sh -listEnabledProfileAll`
- Change all configured run times to Java 7, 64-bit:
 - `managesdk.sh -enableProfileAll -sdkname 1.7_64 -enableServers`
- Change default configured runtime for future servers that are created in a profile:
 - `managesdk.sh -setNewProfileDefault -sdkname 1.7_64`

© Copyright IBM Corporation 2013

Figure 5-18. Using IBM Java 7

WA5913.0

Notes:

A new command-line tool named “managesdk” is added to the “bin” folder of WebSphere Application Server (note that you must separately install IBM Java 7 before managesdk lists this runtime). Use `managesdk.sh -help` to see a list of all tasks and task parameters.

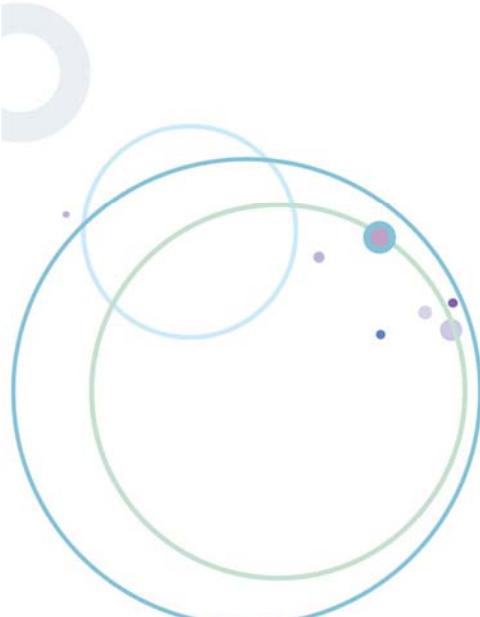
Usage:

```
managesdk.sh -task [ <-parameter> [ <parameter_value> ] | <-parameter> ] .....
task : [ -listAvailable | -listEnabledProfile | -listEnabledProfileAll ]
       -listAvailable [-verbose]
       -listEnabledProfile -profileName <profile_name> [-verbose]
       -listEnabledProfileAll [-verbose]
task : [ -getNewProfileDefault | -setNewProfileDefault ]
       -getNewProfileDefault [-verbose]
       -setNewProfileDefault -sdkName <sdk_name>
```

```
task : [ -getCommandDefault | -setCommandDefault ]
       -getCommandDefault [-verbose]
       -setCommandDefault -sdkName <sdk_name>
task : [ -enableProfile | -enableProfileAll ]
       -enableProfile -sdkName <sdk_name>
                   -profileName <profile_name>
                   [-enableServers]
                   [-user <user_name> -password <password>]
       -enableProfileAll -sdkName <sdk_name>
                   [-enableServers]
task : -help
```


5.2. Introduction to JVM problem determination

Introduction to JVM problem determination



© Copyright IBM Corporation 2013

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

8.0

Figure 5-19. Introduction to JVM problem determination

WA5913.0

Notes:

JVM problem determination: Symptom analysis

- Application server stops responding under the following conditions:
 - Server crash
 - Hung process
 - Out of memory condition
- Server crash
 - Application server stops or exits unexpectedly
 - Tools are available to determine the cause of crash
- Hung process
 - Verify that application server process is still running
 - Threads might be deadlocked
 - Code might be running in a loop
 - Tools available to determine cause of the hang
- Out of memory condition
 - Errors and exceptions that are logged without process exit
 - At times, can result in unexpected process exit
- Performance degradation
 - Application server might crash and the node agent restarts it
 - Check to see whether the process ID is continually changing

© Copyright IBM Corporation 2013

Figure 5-20. JVM problem determination: Symptom analysis

WA5913.0

Notes:

An application server that does not respond might be hung, or the process might be terminated. 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 failing and being restarted automatically.

If CPU activity is low but the application server is 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 an “`OutOfMemoryException`” or just a list of errors and exceptions but no immediate process exit. This situation can occur when the application server is running low on memory, due to an application problem such as a memory leak, a hardware memory failure, or higher than usual demand.

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

JVM problem determination: Data to collect

- Core files
 - Also known as process memory dumps or system core files
 - Complete memory dump of the virtual memory for the process
 - Can be large
 - Usually required by IBM support
 - Tools available to parse these files
- Javacore files
 - Also known as jadump or thread dump files
 - Text file that is created during an application server failure
 - Can also be generated manually
 - Error condition is given at top of memory dump
 - Format of the file is specific to the IBM JDK
- VerboseGC logs
 - Provides detailed information about garbage collection cycles
- Heap memory dump
 - Shows the objects that use Java heap memory
 - Needed for memory leak determination

© Copyright IBM Corporation 2013

Figure 5-21. JVM problem determination: Data to collect

WA5913.0

Notes:

These data artifacts are 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 and 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 that is provided with the specific JVM. On Solaris and HP-UX platforms, the thread dump goes to native_stdout.

Javacore overview

- What is a javacore?
 - A snapshot of the running Java process
 - Small diagnostic text file that the JVM produces contains vital information about the running JVM process
 - Lists the JVM command line, environments, and loaded libraries
 - 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
 - Necessary for detecting hung threads and deadlock conditions
 - Useful for detecting some categories of native memory leaks
- Also, helpful for detecting performance problems
 - Take at least three snapshots of the JVM (about 2 – 3 minutes apart)
 - Analyze the javacores 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 the same monitor or resource might indicate a bottleneck, hang, or a deadlock

© Copyright IBM Corporation 2013

Figure 5-22. Javacore overview

WA5913.0

Notes:

A javacore file is a snapshot of a running Java process. The IBM Java SDK produces a javacore file in response to specific operating system signals. Javacore files show the state of every thread in the Java process, in addition to the monitor information that identifies Java synchronization locks. If deadlocks are detected, it is 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 feature is new for JDK 5.

You can use javacore files to resolve the problems that are listed here. More information on capturing data for problem determination can be found in the MustGather documents for these problems:

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

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

Javacore file location and naming

- The javacore file is stored in the first viable location of:
 - Xdump:java:file=<path_name>/<filename.txt> (command line argument)
 - The setting of the IBM_JAVACOREDIR environment variable (deprecated)
 - <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
 - A new option -Xdump:java:defaults:file=<path/filename> can be used to change the default path and name of all javacore dump agents
- 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

© Copyright IBM Corporation 2013

Figure 5-23. Javacore file location and naming

WA5913.0

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.

If security is enabled or the default SOAP ports are changed, you need to pass extra parameters to the batch file to get a wsadmin prompt. For example: wsadmin.bat [-host host_name] [-port port_number] [-user userid[-password password]]

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

2. Get a handle to the problem application server. (Note the contents in brackets "[...]", along with the brackets, are not optional. It must be entered to set the JVM object. Also, notice 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 deployment manager and if the server names in the cell are not unique, then you can qualify the JVM with a node attribute in addition to process.

1. 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 of the options for controlling javacores, such as naming and location.

Javacore file subcomponents

- **TITLE:** Shows basic information about the event that caused the generation of the javacore, the time it was taken, and the file name
- **GPINFO:** Shows general information about the OS
 - If the memory dump resulted from a general protection fault (GPF), information about the fault module is provided
- **ENVINFO:** Shows information about the JRE level, details about the command line that started the JVM process, and the JVM environment
- **NATIVEMEMINFO:** Provides information about the native memory that the Java Runtime Environment (JRE) allocates
 - Requested from the OS by using library functions such as `malloc()` and `mmap()`
- **MEMINFO:** Shows the free space in heap, the size of current heap, details on other internal memory that the JVM is using, and garbage collection history data
- **LOCKS:** Shows the locks threads hold on resources, including other threads
 - A lock (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 about the class loaders and specific classes loaded

© Copyright IBM Corporation 2013

Figure 5-24. Javacore file subcomponents

WA5913.0

Notes:

GPF stands for general protection fault.

Use the Java command line to determine whether 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** show the Java and native stacks of the current thread. On stacks, the most recent calls are at the top of the stack.

- The Java stack shows the Java method calls that the current thread made.
- If there is a native stack, this javacore contains the most recent events that the thread ran. As the name implies, these events are 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, 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. The signal 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 different than if the javacore is due to a JVM crash.

Each line of information is identified from tags, such as `TISIGINFO`. Different sections contain different tags, which make the file easier to parse for 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` Classloader
 - `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 that the line is just to aid readability.
 - A tag of `SECTION` with the section title heads every section.

To verify that the javacore file is complete, you should see an `END OF DUMP` string at the end of the file.

Javacore example (JDK v6)

```

0SECTION      TITLE subcomponent dump routine
NULL
1TICHARSET    =====
1TISIGINFO    1252
1TIDATETIME   Dump Requested By User (00100000) Through com.ibm.jvm.Dump.JavaDump
1TIFILENAME   Date:          2012/03/06 at 17:08:36
               Javacore filename: C:\ProgramFiles\IBM\WebSphere\AppServer\profiles\
                           profile1\javacore.20120306.170836.6404.0001.txt
1TIREQFLAGS   Request Flags: 0x81 (exclusive+preempt)
1TIPREPSTATE  Prep State:   0x106 (vm_access+exclusive_vm_access+)
NULL
0SECTION      GPINFO subcomponent dump routine
NULL
2XHOSLEVEL   OS Level       : Windows XP 5.1 build 2600 Service Pack 3
2XHCPUUS     Processors - 
3XHCPUARCH   Architecture  : x86
3XHNUMCPUS   How Many      : 1
3XHNUMASUP   NUMA is either not supported or has been disabled by user
1XERROR2     Register dump section only produced for SIGSEGV, SIGILL or SIGFPE.
NULL
0SECTION      ENVINFO subcomponent dump routine
NULL
1CIJAVAVERSION JRE 1.6.0 Windows XP x86-32 build 20111113_94967 (pwi3260_26sr1-
20111114_01(SR1))
1CIVMVERSION  VM build R26_Java626_SR1_20111113_1649_B94967
1CIJITVERSION r11_20111028_21230
1CIGCVERSION  GC - R26_Java626_SR1_20111113_1649_B94967
1CIJITMODES   JIT enabled, AOT disabled, FSD disabled, HCR disabled
1CIRUNNINGAS  Running as a standalone JVM

```

© Copyright IBM Corporation 2013

Figure 5-25. Javacore example (JDK v6)

WA5913.0

Notes:

NUMA stands for non-uniform memory access.

What does non-uniform memory access really mean to you?

Some areas of memory take longer to access because they are different physical buses. More information is available at: <http://lse.sourceforge.net/numa/faq/index.html>

Ahead-of-time (AOT) compilation allows the compilation of Java classes into native code for subsequent executions of the same program. The AOT compiler works with the class data sharing framework.

Verbose garbage collection (GC)

- Verbose GC is an option that the JVM run time provides
- Provides a garbage collection log:
 - Interval between collections
 - Duration of collection
 - Whether compaction was required
 - Memory size, memory that was freed, memory available
- Select **Servers > Server Types > WebSphere application servers > <serverName>**
- Under Server Infrastructure, click **Java and Process Management > Process Definition > Java Virtual Machine**
 - On **Configuration** tab, select **Verbose Garbage Collection** check box, restart server
 - On **Runtime** tab, select **Verbose Garbage Collection** check box, effective for current server instance
- IBM JVM writes to `native_stderr.log`

© Copyright IBM Corporation 2013

Figure 5-26. Verbose garbage collection (GC)

WA5913.0

Notes:

It is often a good idea to have verbose GC enabled permanently in production. The performance cost on a reasonably well-tuned JVM is 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, as a way to monitor the health of the system, even when nothing bad is noticed.

Enabling verbose GC by default is a decision that each system administrator must make consciously. 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 from the **Runtime** tab. This setting allows the initiation of verbose GC output on a running application server.

5.3. JVM tuning

JVM tuning



© Copyright IBM Corporation 2013

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

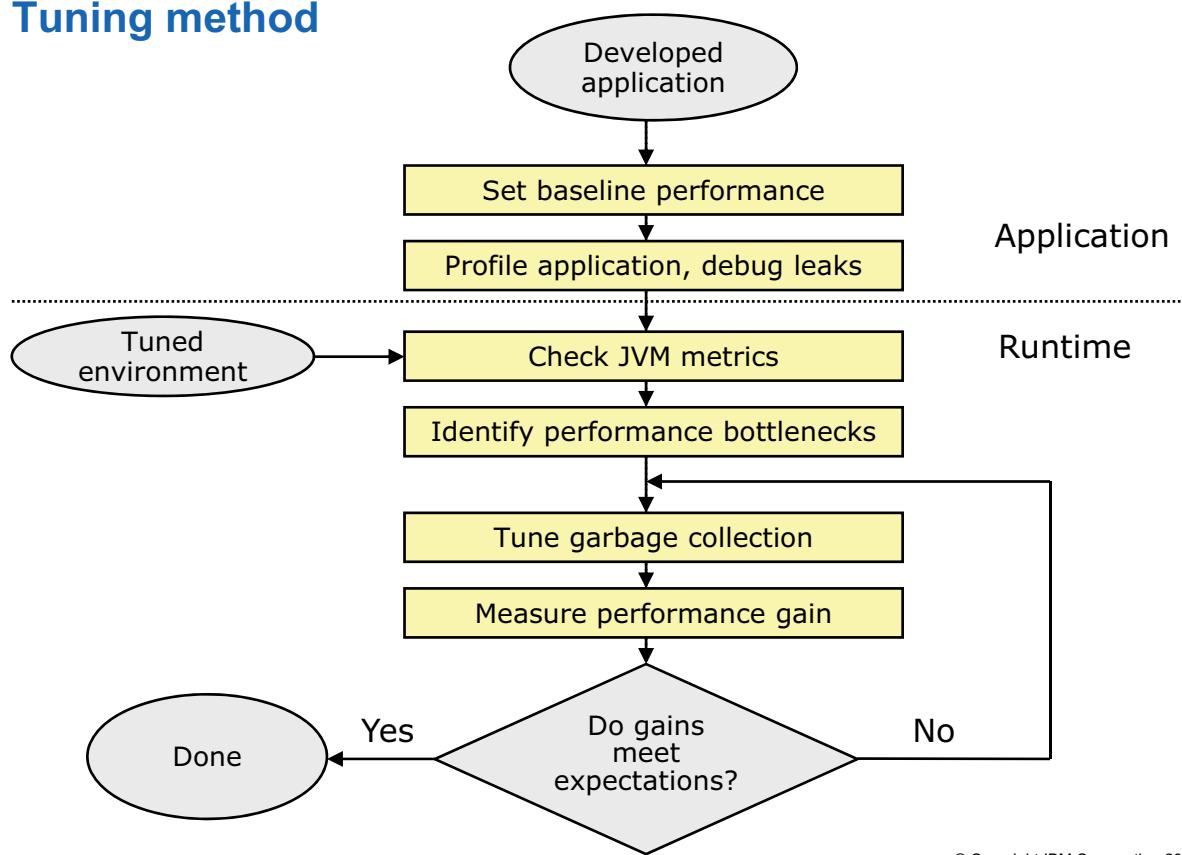
8.0

Figure 5-27. JVM tuning

WA5913.0

Notes:

Tuning method



© Copyright IBM Corporation 2013

Figure 5-28. Tuning method

WA5913.0

Notes:

The tuning process is iterative in nature. To find the optimal configuration, several tests and evaluation should be done.

There are many tuning parameters available to control the behavior of the JVM. These parameters are described detail in the JDK Diagnostic Guide. It is worth noting that most of JVM tuning involves configuring the ideal heap size and to a lesser degree the GC policy. Developers are encouraged to concentrate on these two areas. The other parameters provide fine-tuning and are beyond the scope of this class.

Maximum heap size

- 32-bit Java processes have maximum heap size
 - Varies according to the OS and hardware that is used
 - Determined by the process memory layout
- 64-bit processes do not have this limit
 - Limit exists, but is so large it can be effectively ignored
 - Addressability usually between 2^{44} and 2^{64} which is 16+ terabytes

© Copyright IBM Corporation 2013

Figure 5-29. Maximum heap size

WA5913.0

Notes:

Theoretical and advised maximum heap sizes (1 of 2)

- The larger the Java heap, the more constrained the native heap
- Limits are advised to prevent native heap from becoming overly restricted, leading to OutOfMemoryErrors
- Exceeding advised limits is possible, but should be done only when native heap usage is understood
- Native heap usage can be measured by using OS tools:
 - svmon (AIX)
 - ps (Linux) (for example: `ps -o pid,vsz,rss -p <PID>` where `vsz` is total virtual address space size and `rss` is resident set size)
 - PerfMon (Windows)
 - RMF (z/OS)

© Copyright IBM Corporation 2013

Figure 5-30. Theoretical and advised maximum heap sizes (1 of 2)

WA5913.0

Notes:

The `svmon` command provides a more in-depth analysis of memory usage. It is more informative, but also more intrusive, than the `vmstat` and `ps` commands. The `svmon` command captures a snapshot of the current state of memory. However, it is not a true snapshot because it runs at the user level with interrupts enabled.

The virtual set size (`vsz`) is the total size that the process occupies in virtual memory. The resident set size (`rss`) is the actual amount of space the process occupies in physical memory. Some of the memory that the application requested is now paged out to disk. This paged-out memory is part of the `vsz` number but not the `rss` number.

For example:

```
ps -o pid,vsz,rss -p 19854
  PID    VSZ    RSS
19854  816864  316840
RMF=Resource Measurement Facility
```

Theoretical and advised maximum heap sizes (2 of 2)

Platform	Extra options	Maximum possible	Advised maximum
AIX	automatic	3.25 GB	2.5 GB
Linux		2 GB	1.5 GB
	hugemem kernel	3 GB	2.5 GB
Windows		1.8 GB	1.5 GB
	/3GB	1.8 GB	1.8 GB
z/OS		1.7 GB	1.3 GB

© Copyright IBM Corporation 2013

Figure 5-31. Theoretical and advised maximum heap sizes (2 of 2)

WA5913.0

Notes:

The Linux hugemem kernel supports a 4-GB per process user space (versus 3 GB for the other kernels), and a 4-GB direct kernel space.

The 32-bit IBM JVM for Windows includes support for the `/LARGEADDRESSAWARE` switch, also known as the `/3GB` switch. This switch increases the amount of space available to a process, 2–3 GB. The switch is a Windows boot parameter, not a command-line option to the JVM. After enabling the `/3GB` switch, the JVM gains 1 GB of extra memory space. This extra space does not increase the theoretical maximum size of the Java heap. However, it does allow the Java heap to grow closer to its theoretical maximum size (2 GB - 1 byte) because the extra memory can be used for the native heap.

Tuning considerations: Heap

- Start with a reasonable maximum heap (-Xmx) size
 - 256 MB is the default for an application server
 - Try setting it to 512 MB
- Test different maximum heap sizes to find optimal setting
- Size the maximum heap larger than steady state to allow for peak load
- Consider opposing forces
 - The larger the heap, typically the longer the GC cycle
 - The smaller the heap, the more frequently GC is required
- Setting a larger minimum heap size (-Xms) can improve server startup time
 - If using the 50 MB default, might be resized several times during startup
- Setting the minimum too large can affect runtime performance
 - Larger value means more memory space that requires garbage collection
 - The JVM cannot compensate if you make a poor choice

© Copyright IBM Corporation 2013

Figure 5-32. Tuning considerations: Heap

WA5913.0

Notes:

It is worth noting that with a minimum heap size different from the maximum 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.

The appropriate JVM heap size is usually the only major area of WebSphere Application Server that requires tuning:

- Too little heap:
 - Causes the application to GC frequently
 - Uses too much CPU
 - Extreme cases can lead to error conditions and server failure
- Too much heap:
 - Wasteful
 - Might cause long response times, especially on Solaris

The “correct” Java heap size (1 of 2)

- GC adapts heap size to keep occupancy between 40% and 70%
 - Heap occupancy over 70% causes frequent GC cycles
 - Which generally means reduced performance
- Heap occupancy below 40% means infrequent GC cycles, but cycles longer than they must be
 - Which means longer pause times than necessary
 - Which generally means reduced performance

© Copyright IBM Corporation 2013

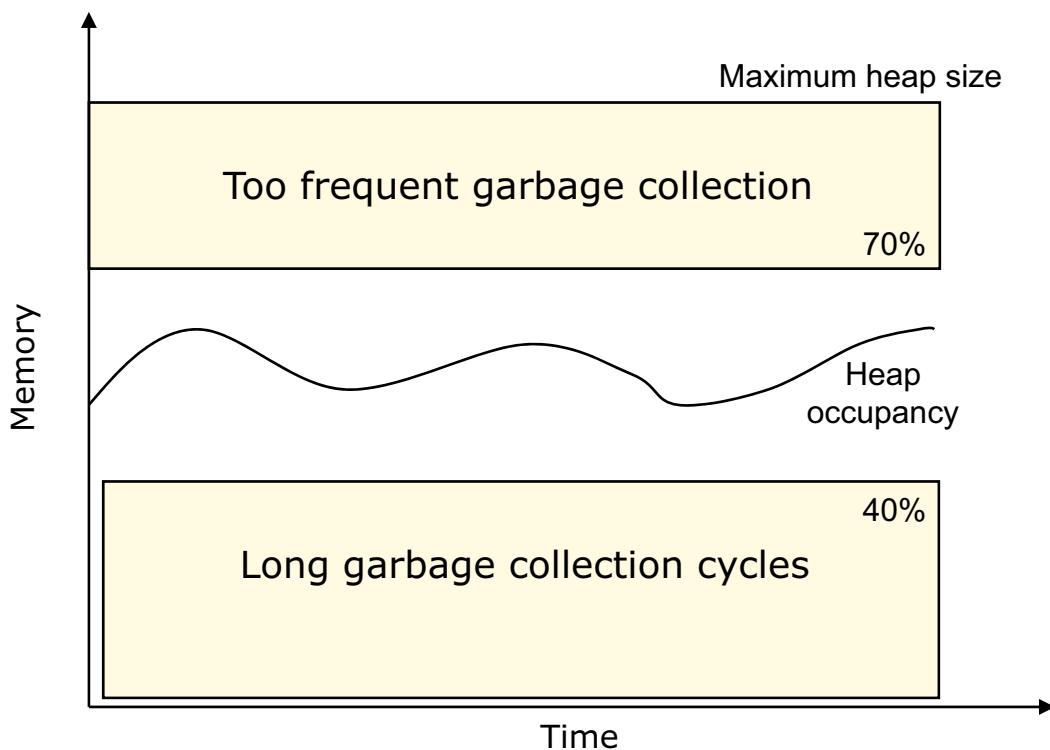
Figure 5-33. The “correct” Java heap size (1 of 2)

WA5913.0

Notes:

Heap occupancy is the amount of live data on the heap.

The "correct" Java heap size (2 of 2)



© Copyright IBM Corporation 2013

Figure 5-34. The "correct" Java heap size (2 of 2)

WA5913.0

Notes:

For most applications, the default settings work well. The heap expands until it reaches a steady state, and then remains in that state, which should give a heap occupancy (the amount of live data on the heap) of 70%. At this level, the frequency and pause time of garbage collection should be acceptable.

Fixed heap sizes versus variable heap sizes

- Should the heap size be “fixed”?
 - That is, minimum heap size (-Xms) = maximum heap size (-Xmx)?
 - Does not expand or shrink the Java heap (avoids compactions)
 - Use for “flat” memory usage
- Variable heap sizes
 - GC adapts heap size to keep occupancy between 40% and 70%
 - Expands and shrinks the Java heap
 - Allows for scenario where usage varies over time, where variations would take usage outside of the 40 – 70% window
 - Provides more flexibility and ability to avoid OutOfMemoryErrors
- Each option has advantages and disadvantages
 - As for most performance tuning, you must select which is right for the particular application

© Copyright IBM Corporation 2013

Figure 5-35. Fixed heap sizes versus variable heap sizes

WA5913.0

Notes:

A common performance-tuning measure is to make the initial heap size (`-Xms`) equal to the maximum heap size (`-Xmx`). Since no heap expansion or contraction occurs, this setting can result in significant performance gains in some situations. Usually, only the applications that need to handle a surge of allocation requests keep a substantial difference between initial and maximum heap sizes. Remember, though, that if `-Xms100m -Xmx100m` is specified, the JVM consumes 100 megabytes of memory for its complete lifetime, even if the heap usage never exceeds 10%.

Tuning considerations: Garbage collection

- “Throughput” (the JVM definition)
 - Measure of productive time
 - Time that is spent in GC is not included
- Pauses
 - Measure of time when application execution pauses during GC
- Turn on verbose GC for more information about GC operation
- Allow the JVM to determine optimum time to GC, 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
 - Can be disabled in IBM JDKs using `-Xdisableexplicitgc`
 - `-Xdisableexplicitgc` must not be a permanent setting as the JVM uses `System.gc()` calls in extreme situations

© Copyright IBM Corporation 2013

Figure 5-36. Tuning considerations: Garbage collection

WA5913.0

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 that is not spent in garbage collection, which is considered over long periods of time. Throughput includes time that is 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 can be tolerable, or network latencies can obscure them. However, in an interactive graphics program even short pauses negatively affect the user experience.

Some users are sensitive to other considerations. *Footprint* is the working set of a process, which is measured in pages and cache lines. On systems with limited physical memory or many processes, footprint can 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).

`-Xdisableexplicitgc` must not be a permanent setting as JVM uses `System.gc` calls in extreme situations (for example, `DirectByteBuffer` native OOM) and this setting inhibits this self-correction.

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
- “I’m using a 64-bit system and need heap sizes greater than 4 GB”
-Xgcpolicy:balanced

© Copyright IBM Corporation 2013

Figure 5-37. Tuning considerations: GC policy

WA5913.0

Notes:

The *subpool* option is deprecated in version 8.

The *balanced* option is intended for environments where heap sizes are greater than 4 GB. The policy is available only on 64-bit platforms.

Tuning considerations: Native heap

- Beware: Garbage collection does not look into the native heap
- Ensure that JVM is not being swapped out of memory to disk
- Total memory that JVM uses > maximum heap size
- Native memory that is 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 that the JVM uses
 - JVMs do not page effectively due to garbage collection

© Copyright IBM Corporation 2013

Figure 5-38. Tuning considerations: Native heap

WA5913.0

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 with the native protocol of the database

JVM tool overview

- Assess the status of a running Java application
 - IBM Monitoring and Diagnostic Tools for Java – Health Center
 - The Health Center can also be used to monitor processor usage and lock contention
- WebSphere internal thread pools and heap usage statistics
 - Tivoli Performance Viewer
- Thread activity snapshot: Use javacore files
 - IBM Thread and Monitor Dump Analyzer for Java
- Memory and garbage collection: Use verbose GC output
 - IBM Monitoring and Diagnostic Tools for Java – Garbage Collection and Memory Visualizer (GCMV)
 - IBM Pattern Modeling and Analysis tool for Java Garbage Collector
- Memory use by object: Use heap dumps
 - Memory Analyzer tool (MAT)
 - IBM HeapAnalyzer

© Copyright IBM Corporation 2013

Figure 5-39. JVM tool overview

WA5913.0

Notes:

Most tuning and debugging can be accomplished with free tools that many customers have on hand or ones available using IBM Support Assistant. In addition, high-end tools can also be used, such as IBM Tivoli Composite Application Manager for WebSphere or other third-party products.

There are tools available to parse system core files.

Most of these tools are covered in more detail in subsequent units.

Unit summary

Having completed this unit, you should be able to:

- Describe components of JVM and overall architecture (Java Development Kit V6.0)
- Describe garbage collection (GC) and GC tuning policies
- Describe JVM command line arguments
- Describe javacore files and how to obtain them
- Identify a sluggish JVM and detect bottleneck problems
- Explain how to tune the heap size
- Use JVM-related tools: Garbage Collection and Memory Visualizer (GCMV), Memory Analyzer tool (MAT), and Java Health Center

© Copyright IBM Corporation 2013

Figure 5-40. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. True or false: In addition to a mark and sweep, a compaction occurs for every garbage collection cycle.
2. True or false: The gencon garbage collection policy is enabled by default for all application servers.
3. True or false: In general, the larger the heap size, the longer the pause time during garbage collection.

© Copyright IBM Corporation 2013

Figure 5-41. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

1. True or false? Explain.

2. True or false? Explain.

3. True or false? Explain.

Checkpoint answers

1. False: A compaction occurs only if the mark and sweep phases cannot recover enough heap space.
2. True: Starting with version 8, gencon policy is the default.
3. True

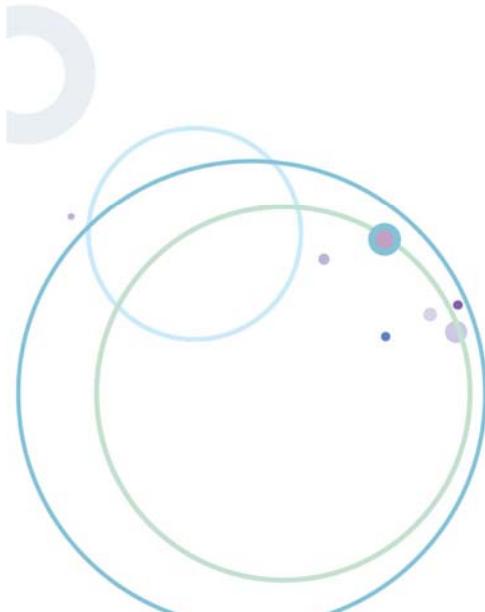
© Copyright IBM Corporation 2013

Figure 5-42. Checkpoint answers

WA5913.0

Notes:

Exercise 3



Introduction to configuring garbage collection policies

© Copyright IBM Corporation 2013

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

8.0

Figure 5-43. Exercise 3

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Enable verbose GC for an application server
- Manually view verbose GC logs
- Configure the optthruput GC policy
- Configure the gencon GC policy
- Use IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) to analyze verbose GC data
- Configure and use the Java Health Center

© Copyright IBM Corporation 2013

Figure 5-44. Exercise objectives

WA5913.0

Notes:

Unit 6. How to troubleshoot hangs

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:

- Describe what a hang is
- Detect a hang condition
- Trigger and analyze javacore files for hangs
- Use the WebSphere Application Server hang detection facility
- Use the IBM Thread and Monitor Dump Analyzer for Java

How you will check your progress

- Checkpoint questions
- Lab exercises

References

SG24-6798-00 WebSphere Application Server V6 Problem Determination
for Distributed Platforms

Unit objectives

After completing this unit, you should be able to:

- Describe what a hang is
- Detect a hang condition
- Trigger and analyze jvavacore files for hung threads
- Use the WebSphere Application Server hang detection facility
- Use the IBM Thread and Monitor Dump Analyzer for Java

© Copyright IBM Corporation 2013

Figure 6-1. Unit objectives

WA5913.0

Notes:



Topics

- Application server hangs
- Hung thread detection
- IBM Thread and Monitor Dump Analyzer

© Copyright IBM Corporation 2013

Figure 6-2. Topics

WA5913.0

Notes:

6.1. Application server hangs

Application server hangs



© Copyright IBM Corporation 2013

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

8.0

Figure 6-3. Application server hangs

WA5913.0

Notes:

Application server hangs defined

Clarify the nature and extent of the hang

- A “hang” is not the same as a “crash”
- Is the entire process truly hung, or does it still respond to some requests?
 - Test with sample “Snoop” servlet, wsadmin commands, and other commands
- Deadlocks:
 - Often, one process fails to respond to a request because it called another process that is itself hung; sometimes it is hard to find the true culprit
 - Deadlocks also can occur within the same Java process, where one thread is deadlocked on another

© Copyright IBM Corporation 2013

Figure 6-4. Application server hangs defined

WA5913.0

Notes:

This hung transaction might consume system resources, such as processor time, when threads run unbounded code paths, such as when the code is running in an infinite loop. Alternatively, 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 can remain in this degraded state indefinitely.

Notice that some deadlocks or other simpler hangs are contained entirely within one process; all the threads and monitors that are involved in the same JVM process and are visible in a single javacore. Other deadlocks can 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 can 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 can see a bunch of threads that made some remote request and are waiting for responses. You must 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.

Thread hangs

- WebSphere Application Server threads can become hung due to running poorly coded Java EE applications
- WebSphere administrators might not know that 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

© Copyright IBM Corporation 2013

Figure 6-5. Thread hangs

WA5913.0

Notes:

There are tools available, and WebSphere has a built-in monitor to help identifying hung threads. These tools and the monitor are reviewed later in the unit. Using these facilities, it can be a simple process to identify when threads are hung.

Keep in mind that not all hangs are the result of application coding problems. They can be the result of problems in the JVM with WebSphere.

Thread hang examples in application code

- Understand root causes
 - Customer code enters an endless 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
    }
}
```

© Copyright IBM Corporation 2013

Figure 6-6. Thread hang examples in application code

WA5913.0

Notes:

Here are some examples of coding practices that lead to hung threads. Such coding errors are typically discovered and fixed during the development and testing phase of an application.

WebSphere process hangs detection steps

1. Collect OS statistics, processor usage by other processes, and virtual memory paging
 - The JVM might be hung because OS resources are exhausted
2. When hung threads are suspected, manually trigger a thread dump
 - Use wsadmin or OS facilities; see next slide
 - Create a script or use script in IBM Support Assistant to collect MustGather data when the process that is suspected hangs
 - Distinguish the 100% CPU cases from idle CPU cases
3. For a typical hang, collect three javacore dumps a few minutes apart
 - To see whether anything is moving within the process (but slowly)
4. Examine the thread dumps manually or with tools
 - Look for deadlocks
 - Look for large number of threads that are blocked
 - Look for threads that are waiting after sending a request to some other process, now awaiting a response

© Copyright IBM Corporation 2013

Figure 6-7. WebSphere process hangs detection steps

WA5913.0

Notes:

Java dumps are enabled by default. Java dump production can be turned off using `-Xdump:java:none`. Disabling a Java dump is a bad idea because Java dumps are essential diagnostics tools. Use the `-Xdump:java` option to give more fine-grained control over the production of Java dumps with IBM JDK on AIX, Linux, or Windows.

Note: The use of `-Xdump` is introduced in IBM JDK 5. For earlier versions of the IBM JDK the Java dump can be disabled by setting the `DISABLE_JAVADUMP` environment variable to TRUE.

By default, a Java dump is triggered when one of the following occurs:

- An unrecoverable 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 run.

The exact conditions in which you get a Java memory dump vary depending on whether the default memory dump agents are overridden. An unrecoverable exception is one that causes the JVM to

fail. The JVM handles this situation by producing a `javacore` file and then stopping the process. In the user-controlled cases (the latter two), the JVM pauses, creates the memory dump, and then continues execution.

Later you examine two tools that can help in analyzing the thread memory dump.

How to manually trigger a thread dump

- Use operating system facilities:
 - `kill -3 <WAS PID>` (UNIX or Linux)
- Use the Java core button in the administrative console:
 - Click **Troubleshooting > Java dumps and cores > server_name > Java core**
- Explicitly tell WebSphere to generate a thread dump
 - From a command line, start the wsadmin shell
 - Specify `-lang jython` on the command line
 - Run the following Jython commands:

```
jvm = AdminControl.completeObjectName(
    'type=JVM,process=<server_name>,*' )
AdminControl.invoke(jvm, 'dumpThreads')
```

© Copyright IBM Corporation 2013

Figure 6-8. How to manually trigger a thread dump

WA5913.0

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. For Solaris or HP-UX JDK, a thread dump is printed in `native_stdout.log`.

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 are changed, you must pass more parameters to the batch file to get a wsadmin prompt. For example:

```
wsadmin.bat [-host host_name] [-port port_number] [-lang jacl/jython] [-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 to which it is attached. Depending on the process to which it is attached, you can get thread dumps for various JVMs. If wsadmin is connected to the 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 is connected.

2. Get a handle to the problem application server. (Note: the contents in brackets “[...]”, along with the brackets, are not optional. It must be entered to set the JVM object. 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 deployment manager and if the server names in the cell are not unique, then you can qualify the JVM with a node attribute in addition to the process.

3. Generate the thread dump:

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

4. You can keep the wsadmin session and use the `$AdminControl invoke $jvm dumpThreads` command multiple times to generate multiple thread dumps.
5. For Jython commands, replace steps 2 and 3 with the following commands:

```
jvm = AdminControl.completeObjectName('type=JVM, process=server1,*')
AdminControl.invoke(jvm, 'dumpThreads')
```

Thread dump analysis

What are the significant pictures in the thread dump?

- Many threads are waiting in the same method for some resource
 - Probably a synchronization issue
 - Might be an outage or slow response from remote server
- No activity
 - WebSphere Application Server is not receiving traffic for some reason
 - Check front-end resources, networks, test clients, and so forth
 - Also, check timing of the thread dump
- Hundreds of threads
 - Shared resource not available
 - Customer must control web container threads better
 - Might need more capacity

© Copyright IBM Corporation 2013

Figure 6-9. Thread dump analysis

WA5913.0

Notes:

Thread dumps are snapshot of the running JVM. They vary in format and detail among operating systems.

IBM JDK generates a separate javacore file. Solaris and HP-UX JDK print thread dumps in the `native_stdout.log` file.

Javacore hang indicators

- Look for the string “Deadlock detected”
- JVM monitor information:
 - Shows synchronization locks
 - Indicates blocked threads
- Active threads
 - Look for state:R, which indicates runnable threads
 - This example shows that the thread is doing I/O
 - If this thread is doing the same operation across multiple javacore files, there might be a network interface issue

```
"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))
```

© Copyright IBM Corporation 2013

Figure 6-10. Javacore hang indicators

WA5913.0

Notes:

The monitor information shows what synchronization locks are held by which threads. It also shows which threads the monitors are blocking. 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.

Many blocked threads on a monitor do not mean that a deadlock occurred. It might mean that a monitor (synchronization lock) 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 whether the thread is 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, it might be because:

- A `sleep()` call is made.
 - The thread is blocked for I/O.
 - A synchronized method of an object that is locked by another thread has been called.
 - The thread is synchronizing with another thread with a `join()` call.
- S – suspended: The thread is suspended by another thread.
 - Z – zombie: The thread is killed.
 - P – parked: The new concurrency API (`java.util.concurrent`) parked the thread.
 - 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. The stack shows the Java calls that are 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 be called. It 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.

Thread state values

- 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, the thread might be waiting because:
 - A `sleep()` call is made
 - The thread is blocked for I/O
 - A synchronized method of an object that is locked by another thread was called
 - The thread is synchronizing with another thread with a `join()` call
- S – suspended: The thread suspends another thread
- Z – zombie: The thread was killed
- P – parked: The new concurrency API (`java.util.concurrent`) parks thread
- B – blocked: The thread is waiting to obtain a lock that something else currently owns

© Copyright IBM Corporation 2013

Figure 6-11. Thread state values

WA5913.0

Notes:

Thread states: The thread state indicates whether the thread is 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.

Java 7: More details about monitor usage

- The Java stack trace might include extra lines that begin with 5XESTACKTRACE
 - Shows the locks that were taken in the method on the previous line in the trace
 - Shows a cumulative total of how many times the locks were taken within that stack at that point
 - This information is useful for determining the locks that are held by a thread, and when those locks are released
- Example Java stack trace for a thread that calls `java.io.PrintStream` methods:

```
5XESTACKTRACE at java/io/PrintStream.write(PrintStream.java:504 (Compiled Code))
5XESTACKTRACE (entered lock: java/io/PrintStream@0xA1960698, entry count: 3)
5XESTACKTRACE at
sun/nio/cs/StreamEncoder.writeBytes(StreamEncoder.java:233 (Compiled Code))
5XESTACKTRACE at
sun/nio/cs/StreamEncoder.implFlushBuffer(StreamEncoder.java:303 (Compiled Code))
5XESTACKTRACE at
sun/nio/cs/StreamEncoder.flushBuffer(StreamEncoder.java:116 (Compiled Code))
5XESTACKTRACE (entered lock:java/io/OutputStreamWriter@0xA19612D8, entry count: 1)
```

© Copyright IBM Corporation 2013

Figure 6-12. Java 7: More details about monitor usage

WA5913.0

Notes:

The Java stack trace includes information about locks that were taken within that stack by calls to synchronized methods or the use of the synchronized keyword.

After each stack frame in which one or more locks were taken, the Java stack trace might include extra lines that start with 5XESTACKTRACE. These lines show the locks that were taken in the method on the previous line in the trace, and a cumulative total of how many times the locks were taken within that stack at that point. This information is useful for determining the locks that are held by a thread, and when those locks are released.

Java locks are re-entrant; they can be entered more than once. If a method contains multiple occurrences of the synchronized keyword, the result can be that the same lock is entered more than once in that method. Because of this behavior, the entry counts might increase by more than one, between two method calls in the Java stack, and a lock might be entered at multiple positions in the stack. The lock is not released until the first entry, the one furthest down the stack, is released.

Java locks are released when the `Object.wait()` method is called. Therefore, a record of a thread that entered a lock in its stack does not guarantee that the thread still holds the lock. The thread might be waiting to be notified about the lock, or it might be blocked while attempting to

reenter the lock after being notified. In particular, if another thread calls the `Object.notifyAll()` method, all threads that are waiting for that monitor must compete to reenter it, and some threads become blocked. You can determine whether a thread is blocked or waiting on a lock by looking at the `3XMTHREADBLOCK` line for that thread. A thread that calls the `Object.wait()` method releases the lock only for the object on which it called the `Object.wait()` method. All other locks that the thread entered are still held by that thread.

See the information center topic “Understanding Java and native thread details”:

http://pic.dhe.ibm.com/infocenter/java7sdk/v7r0/topic/com.ibm.java.lnx.70.doc/diag/tools/javadump_tags_javaandnative_thread_detail.html

Javacore hang symptoms (1 of 2)

- Check to see whether threads are blocked waiting on monitors
 - Might indicate bottleneck on unavailable resources or poor synchronization logic
 - Deadlocks within the process are noted in the javacore
- If threads are in a running state
 - Check method across multiple javacore files
 - If individual threads in the same method, might indicate looping logic
- If threads are in wait states, might indicate that a resource (local or remote) is causing the hang
- Technical note:
 - Some threads that are Runnable, are shown as in a Conditional Wait
 - Since IBM JVM version 5, when a javacore is taken, some threads in a Runnable (R) state enter Conditional Wait (CW) state during the javacore
 - This behavior is by design and it is meant to maximize internal consistency during the process of creating the javacore
 - This behavior increases the quality of the javacore and lessens the potential for a crash or data corruption

© Copyright IBM Corporation 2013

Figure 6-13. Javacore hang symptoms (1 of 2)

WA5913.0

Notes:

Threads waiting on monitors are not usually deadlocks. Most of them are bottleneck or synchronization issues where the active thread is in some type of long timeout or resource shortage issue.

You must follow a chain of blocked threads to get to the root cause. For example, thread A might block multiple threads, while thread B blocks thread A, and thread B is waiting for a response from a remote server.

Java cores contain much information and cover dozens of threads. It is good to use tools to process the javacore, such as the Thread and Monitor Dump Analyzer. If such a tool is not available, important information can still 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.

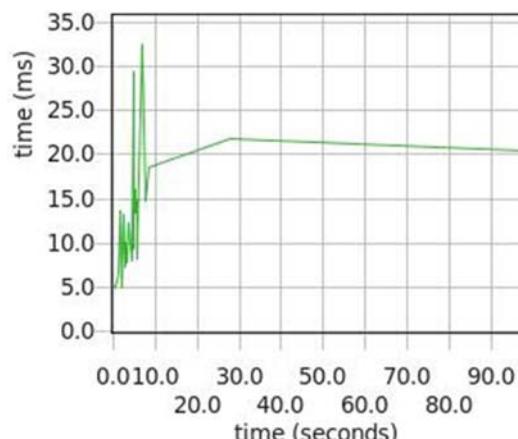
If a hung thread is suspected, also check for log messages from the hang detection facility included with WebSphere.

Javacore hang symptoms (2 of 2)

- The JVM might be hung because it is spending too much time in garbage collection
- An IBM javacore file can be loaded directly into the Garbage Collection and Memory Visualizer tool to review the garbage collection usage for the last few cycles before the javacore was triggered
 - Enable the Total Pause Time statistic in the Memory template

Total pause time

Mean	Minimum	Maximum	Total
time (ms)	time (ms)	time (ms)	time (ms)
12.8	4.88	32.5	372



© Copyright IBM Corporation 2013

Figure 6-14. Javacore hang symptoms (2 of 2)

WA5913.0

Notes:

In general, you would load verbose GC data into the Garbage Collection and Memory Visualizer tool and analyze it. Using the javacore file can help when you are troubleshooting hangs to determine whether excessive garbage collection, rather than hung threads, caused the hang.

Hang detection tools

- **ThreadMonitor:** ThreadMonitor architecture was created to monitor thread pools within WebSphere
- Notification of potentially hung threads is logged
- Monitored pools include:
 - Web container thread pool
 - ORB thread pool
 - Others
- IBM Thread and Monitor Dump Analyzer
 - GUI-based tool
 - Gathers and analyzes thread dumps from a WebSphere Application Server
 - Provides tuning recommendations that are based on analysis
- IBM Monitoring and Diagnostic Tools for Java – Garbage Collection and Memory Visualizer
 - A verbose GC data visualizer
 - Parses and plots various log types that include verbose GC logs, `-Xtgc` output, native memory logs (output from `ps`, `svmon`, and `perfmon`)

© Copyright IBM Corporation 2013

Figure 6-15. Hang detection tools

WA5913.0

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 3 minutes. The performance degradation for this monitoring is less than 1%.

These tools are covered in more details in the following sections.

6.2. Hung thread detection

Hung thread detection



© Copyright IBM Corporation 2013

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

8.0

Figure 6-16. Hung thread detection

WA5913.0

Notes:

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 asynchronous bean thread pools
 - Turned on by default
- Unmanaged threads are not monitored
 - Threads that applications create (not allowed in Java EE)
 - Some internal threads
- Upon notification of a hung thread:
 - Obtain a javacore and see what the thread is doing
 - You can configure WebSphere to generate a javacore automatically when a hung thread is detected
 - Investigate the nature of the thread

© Copyright IBM Corporation 2013

Figure 6-17. WebSphere hung thread detection

WA5913.0

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 doing. Investigate the nature of the thread, the nature of the application, and the frequency of this kind notice. Even try to decide whether this thread hang is normal.

Hung thread detection internals

- When the thread pool gives work to a thread, it notifies the thread monitor
 - Thread monitor notes thread ID and time stamp
- Thread monitor compares active threads to time stamps
 - Threads active longer than the time limit are marked “potentially hung”
- Performance impact is minimal (< 1%)

© Copyright IBM Corporation 2013

Figure 6-18. Hung thread detection internals

WA5913.0

Notes:

When the thread pool dispatches 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%.

Hung thread detection notification

- No action that is taken to kill the thread – only a notification mechanism
 - Thread dump can be taken when hung thread is detected
- If thread is suspected to be hung, notification is sent to:
 - JMX listeners
 - PMI clients for ThreadPool metric (counters are updated)
 - Message that is written to SystemOut.log:

```
[7/10/13 11:51:30:243 EST] 00000020 ThreadMonitor W WSVR0605W: Thread
"WebContainer : 2" (00000028) has been active for 64828 milliseconds
and may be hung. There is/are 1 thread(s) in total in the server that
may be hung.
at java.lang.Thread.sleep(Native Method)
at java.lang.Thread.sleep(Thread.java:850)
at
com.ibm.websphere.samples.plantsbywebspherewar.ShoppingServlet.perfor
mTask(ShoppingServlet.java:141)
. . . [remaining stack traces have been truncated]
```

© Copyright IBM Corporation 2013

Figure 6-19. Hung thread detection notification

WA5913.0

Notes:

The stack trace portion of the message is new in version 7.

The thread monitor does not try to deal with the hung threads; it just sends notifications so that the administrator or developer can deal with the problems.

One can configure WebSphere to generate a thread dump when a hung thread is detected.

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:
 - Sends a Java Management Extensions (JMX) notification. This notification enables third-party tools to catch the event and take appropriate action, such as triggering a JVM thread dump of the server, or send an electronic page or email. The following JMX notification events are defined in the com.ibm.websphere.management.NotificationConstants class:
 - TYPE_THREAD_MONITOR_THREAD_HUNG: The detection of a (potentially) hung thread triggers this.

- 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 might be hung. There are *totalthreads* threads in total in the server that might 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.

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:

```
[7/10/13 11:51:47:210 EST] 00000020 ThreadMonitor W WSVR0605W:
Thread "WebContainer : 2" (00000028) was previously reported to be
hung but has completed. It was active for approximately 65900
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

© Copyright IBM Corporation 2013

Figure 6-20. Hung thread detection false alarms

WA5913.0

Notes:

It is possible that a thread might be running for longer than the specified threshold for legitimate reasons. For example, a thread might be running 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 some false alarms, the threshold automatically increases by 50% to account for these long-running threads. The idea is that if several threads are routinely active for 20 minutes, the threshold eventually adjusts itself to be higher than 20 minutes, to not mark those threads as hung.

Hung thread detection configuration

- Custom properties for hung thread detection configuration
 - Select **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	seconds	180	The interval at which the thread pools are polled for hung threads
com.ibm.websphere.threadmonitor.threshold	seconds	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%
com.ibm.websphere.threadmonitor.dump.java	N/A	False	Set to true to cause a javacore to be created when a hung thread is detected and a WSVR0605W message is printed

© Copyright IBM Corporation 2013

Figure 6-21. Hung thread detection configuration

WA5913.0

Notes:

These custom properties are not present unless added to the configuration. If not present, the default values take effect. To change the behavior, add these properties that provide values that you want.

The hang detection policy can be configured by creating custom properties for the application server.

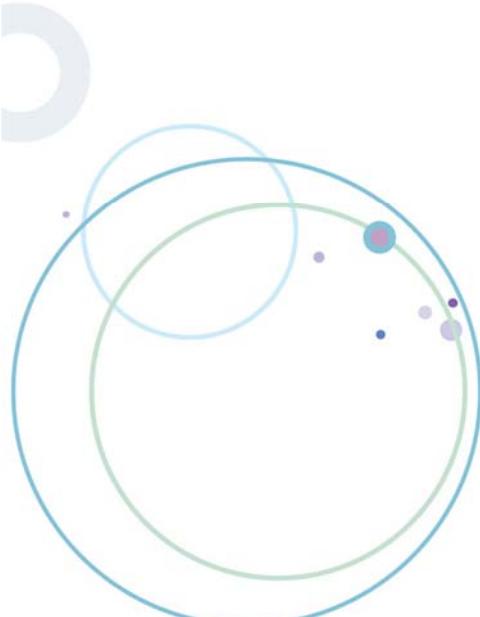
- com.ibm.threadmonitor.interval is the interval at which the thread pools are polled for hung threads (in seconds). It defaults to 180 seconds, which is 3 minutes.
- com.ibm.websphere.threadmonitor.threshold is the length of time that a thread can be active before being marked as "potentially hung". The default value is 10 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.
- com.ibm.websphere.threadmonitor.dump.java. If set to true, javacore is taken when a hung thread is detected. This javacore can be helpful to discover a hung thread in an action automatically.

The application server must be restarted for these changes to take effect. To adjust the hang detection policy proactively, 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.

6.3. IBM Thread and Monitor Dump Analyzer

IBM Thread and Monitor Dump Analyzer



© Copyright IBM Corporation 2013

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

8.0

Figure 6-22. IBM Thread and Monitor Dump Analyzer

WA5913.0

Notes:

What is Thread and Monitor Dump Analyzer (TMDA)?

- TMDA is a tool to analyze thread dumps
 - Available from the IBM Support Assistant
- Used for:
 - Analyzing threads and monitors in javacore files
 - Comparing multiple javacore files from the same process
- Friendlier interface for novice thread dump readers
 - Provides graphical interface to view contents of the thread dump
- Used to analyze threads for:
 - Performance bottlenecks
 - Determining whether deadlocks exist
 - Determining whether threads are being blocked on monitors (might not be a deadlock)

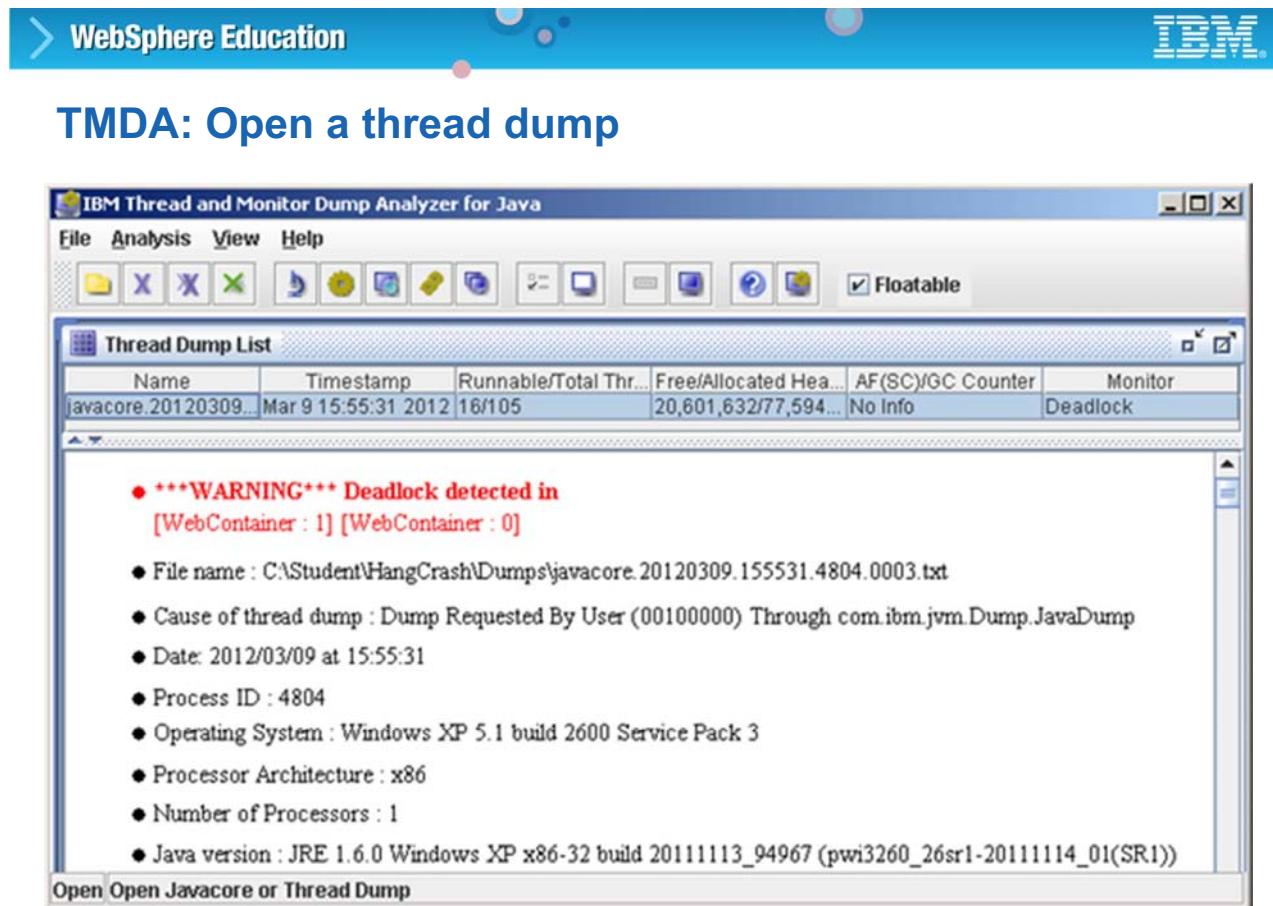
© Copyright IBM Corporation 2013

Figure 6-23. What is Thread and Monitor Dump Analyzer (TMDA)?

WA5913.0

Notes:

The ThreadAnalyzer tool is still available through IBM Support Assistant. It is deprecated and not actively maintained.



© Copyright IBM Corporation 2013

Figure 6-24. TMDA: Open a thread dump

WA5913.0

Notes:

You can search the local file system for one or more javacore files. Each file is loaded into the tool and analyzed.

When a thread dump is loaded and selected, TMDA displays a summary information of the selected thread dump.

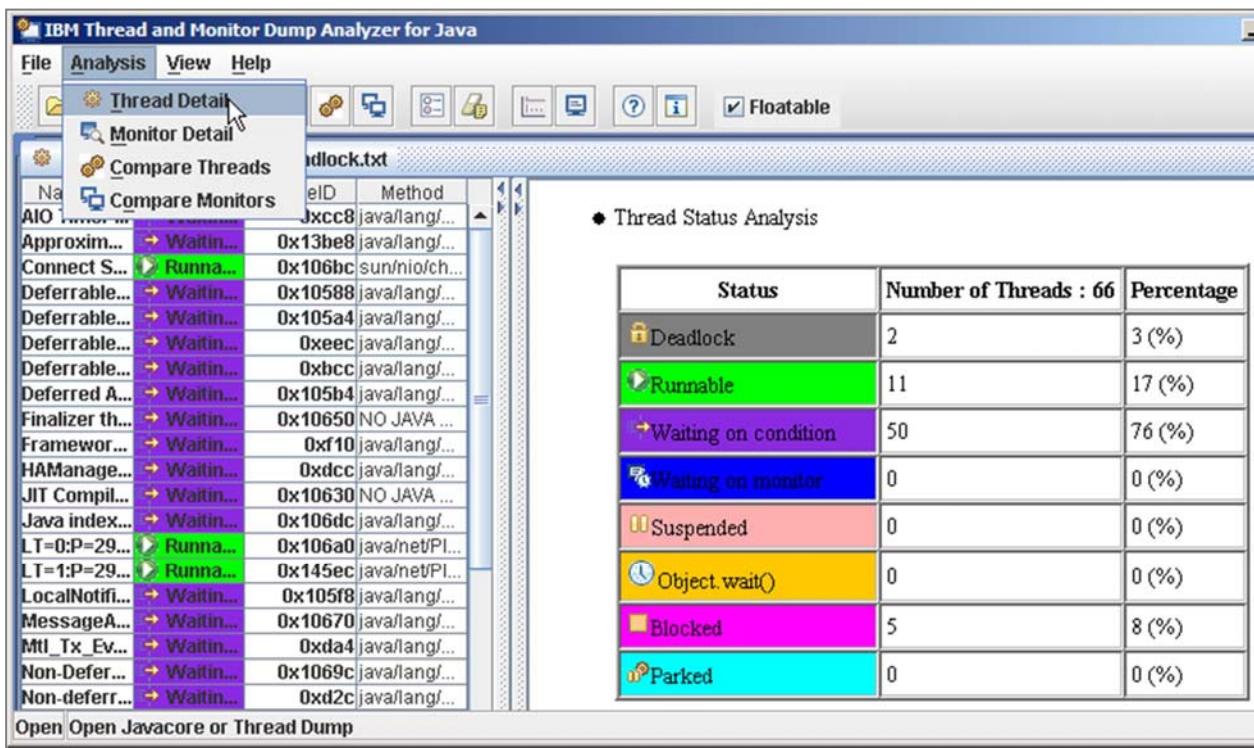
The following information is displayed for IBM javacore:

- Warning if any deadlocked threads are found
- File name
- Cause of thread dump
- Timestamp of the javacore
- Process ID
- Java version
- Java heap information
 - Maximum Java heap size

- Initial Java heap size
- Garbage collector counter
- Allocation failure counter
- Free Java heap size
- Allocated Java heap size
- Memory segment analysis (only for IBM JDK 5 and above)
- Current thread name
- Number of loaded classes in Java heap
- Recommended size of cluster (only applicable to IBM SDK 1.4.2 and 1.3.1 SR7 or later)
- Number of classloaders on the Java heap
- Java command line
- Thread status analysis – group threads by state, such as runnable, waiting
- Thread method analysis – group threads by the current method
- Thread aggregation analysis – group threads by thread function, such as ORB threads or web container threads

For Solaris and HP-UX jvavacore, only the file name, thread status analysis, thread method analysis, and thread aggregation analysis are provided.

Thread detail: Thread status analysis



© Copyright IBM Corporation 2013

Figure 6-25. Thread detail: Thread status analysis

WA5913.0

Notes:

The Analysis menu allows you to display thread and monitor details for a single javacore. If you open multiple javacores, you can display a comparative thread or monitor analysis.

The Thread Detail Analysis displays:

- Thread Status Analysis
- Thread Method Analysis
- Thread Aggregation Analysis

Thread Status Analysis shows the number of threads in each state: deadlocked, runnable, or blocked, for example.

The left pane lists all thread in the thread dump. Threads can be sorted by thread name, state, native ID, or current method.

When you select a thread from the left pane, the right pane shows Thread Detail View. Thread Detail View provides the following information:

- Thread name: The name of a thread
- Thread state: The state of a thread; for example, runnable, waiting, or suspended
- Method name: The latest method that is run or predefined status or stack trace pattern; for example, IDLE, LISTEN, and KEEP-ALIVE
- Java Stack Trace: Java Stack Trace is shown when a thread is selected
- Native Stack Trace: Native Stack Trace is shown after the Java Stack Trace if it is available

The thread states 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 is blocked for I/O
 - A synchronized method of an object that is locked by another thread that was called
 - The thread is synchronizing with another thread with a `join()` call
- S – suspended: Another thread suspended the thread.
- Z – zombie: The thread is killed.
- P – parked: The new concurrency API (`java.util.concurrent`) parked the thread.
- B – blocked: The thread is waiting to obtain a lock that something else currently owns.

Note:

- An IDLE thread is a thread that is ready to receive work but does not have a connection that is established.
- A KEEP-ALIVE thread is an idle thread that is ready to receive work and does have a connection that is established.
- A LISTEN thread listens on a port.

Thread detail: Thread method analysis

Method Name	Number of Threads : 66	Percentage
java/lang/Object.wait(Native Method)	42	64 (%)
java/lang/Thread.sleep(Native Method)	7	11 (%)
java/net/PlainSocketImpl.socketAccept(Native Method)	3	5 (%)
sun/nio/ch/WindowsSelectorImpl\$SubSelector.poll0(Native Method)	2	3 (%)
com/ibm/issf/atjolin/badapp/BadAppServlet.sneezyMethod(BadAppServlet.java:332)	2	3 (%)
com/ibm/io/async/AsyncLibrary.aio_getioev2(Native Method)	2	3 (%)
NO JAVA STACK	2	3 (%)
com/ibm/jvm/Dump.JavaDump(Native Method)	1	2 (%)
com/ibm/issf/atjolin/badapp/BadAppServlet.sneezyMethod(BadAppServlet.java:337)	1	2 (%)
com/ibm/issf/atjolin/badapp/BadAppServlet.dopeyMethod(BadAppServlet.java:320)	1	2 (%)
java/net/PlainDatagramSocketImpl.receive0(Native Method)	1	2 (%)
java/net/SocketInputStream.socketRead0(Native Method)	1	2 (%)
com/ibm/misc/SignalDispatcher.waitForSignal(Native Method)	1	2 (%)

© Copyright IBM Corporation 2013

Figure 6-26. Thread detail: Thread method analysis

WA5913.0

Notes:

This table groups threads by the current threads. It is a convenient view to find out whether many threads are blocked in the same method.

Thread detail: Thread aggregation analysis

Thread Type	Number of Threads : 66	Percentage
Thread	11	17 (%)
Alarm	6	9 (%)
WebContainer	5	8 (%)
Deferrable Alarm	4	6 (%)
SoapConnectorThreadPool	3	5 (%)
ThreadManager.JobsProcessorThread.InternalThread	1	2 (%)
WLMMonitorSleeper	1	2 (%)
ServerSocket	1	2 (%)
HAManager.thread.pool	1	2 (%)

© Copyright IBM Corporation 2013

Figure 6-27. Thread detail: Thread aggregation analysis

WA5913.0

Notes:

Thread aggregation analysis groups threads by function from the perspective of WebSphere; for example, ORB threads, web container threads.

Thread detail: Memory segment analysis

Memory Type	# of Segments	Used Memory(bytes)	Used Memory(%)	Free Memory(bytes)	Free Memory(%)	Total Memory(bytes)
Internal	102	6,567,172	98.24	117,500	1.76	6,684,672
Object	1	65,131,520	100	0	0	65,131,520
Class	1,090	77,451,880	95.1	3,988,936	4.9	81,440,816
JIT Code Cache	7	0	0	3,670,016	100	3,670,016
JIT Data Cache	5	2,214,476	84.48	406,964	15.52	2,621,440
Overall	1,205	151,365,048	94.87	8,183,416	5.13	159,548,464

© Copyright IBM Corporation 2013

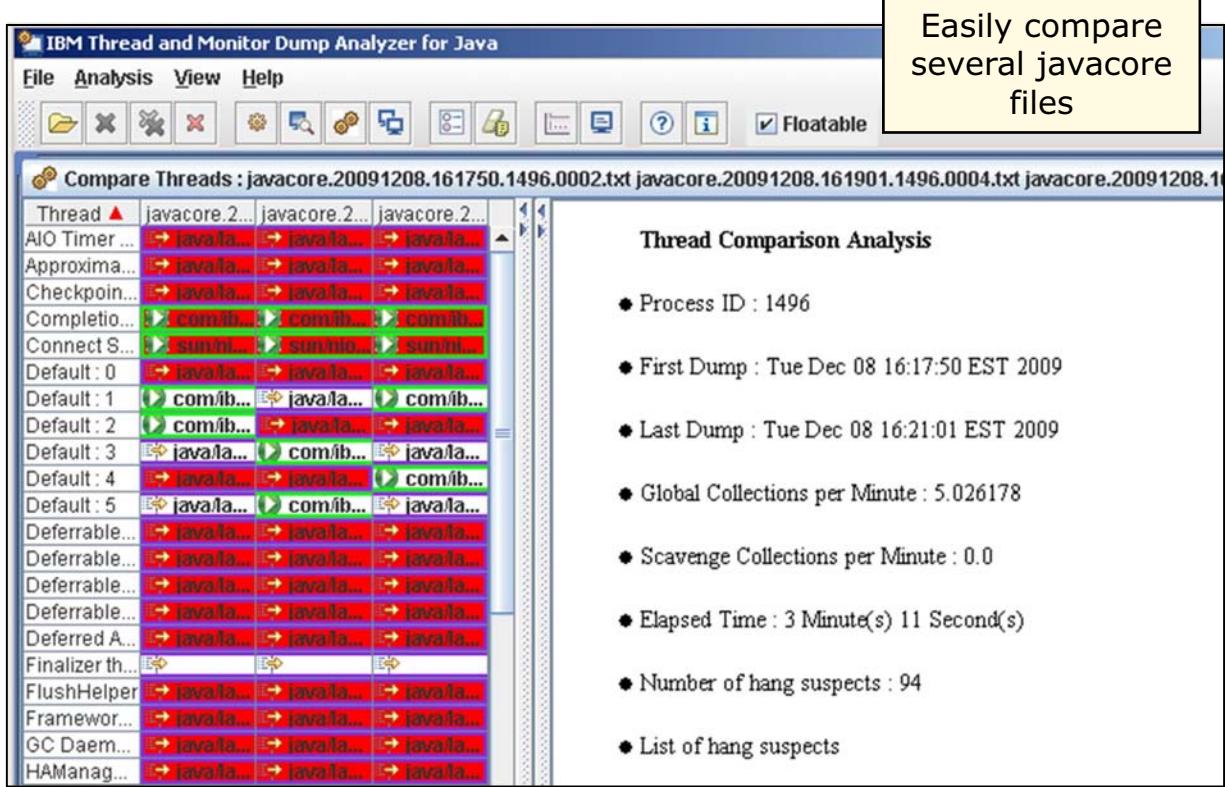
Figure 6-28. Thread detail: Memory segment analysis

WA5913.0

Notes:

The memory segment analysis is available only for IBM JDK 5 and above. It provides information of memory usage by JVM process. It can be useful in debugging native memory problems.

Multiple dump comparative analysis



© Copyright IBM Corporation 2013

Figure 6-29. Multiple dump comparative analysis

WA5913.0

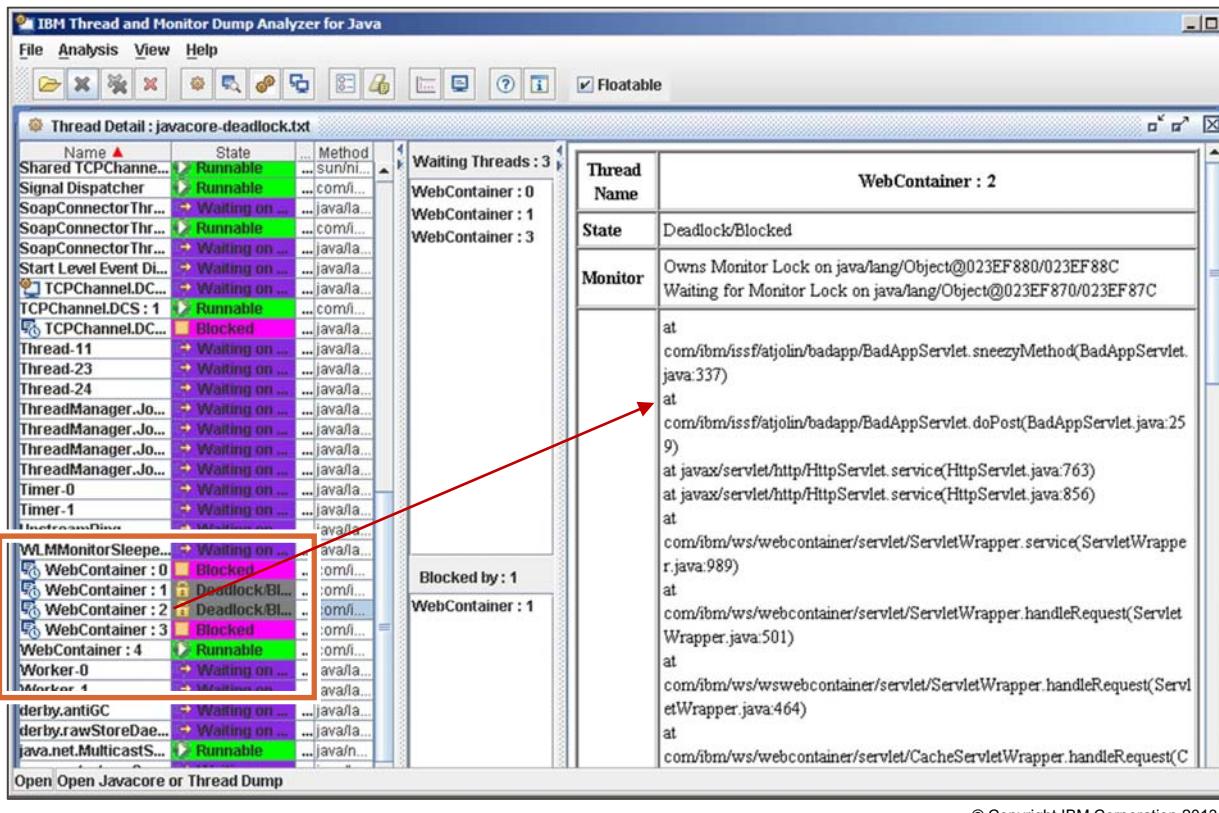
Notes:

The Compare Threads View for two or more javacores consists of the following information:

- Summary is displayed in the right pane. Thread name and method name are displayed from each javacore on the table.
- Process ID.
- First Dump: Timestamp of the first javacore.
- Last Dump: Timestamp of the last javacore.
- Garbage Collections per Minute: Number of garbage collections per minute.
- Allocation Failures per Minute: Number of allocation failures per minute.
- Elapsed Time: Time between the first javacore and the last javacore.
- Number of hang suspects.
- List of hang suspects. Many threads are in idle state and they are hang suspects.



Thread analysis: Deadlocked thread details



© Copyright IBM Corporation 2013

Figure 6-30. Thread analysis: Deadlocked thread details

WA5913.0

Notes:

In the left pane, each thread name can be selected, and the details of the thread are displayed in the right pane.

Notice that in the web container, two threads are selected in the screen capture. The middle pane shows threads that are waiting and the threads that web container 2 is blocking.

Thread analysis: Monitor details

The screenshot shows the 'Monitor Detail' view for a file named 'javacore-deadlock.txt'. On the left, a hierarchical tree displays various threads and their locking status. A specific thread, 'WebContainer : 1', is highlighted in yellow and is shown to be holding a lock on a Java object. This thread is identified as being in a 'Deadlock/Blocked' state, as indicated by a red warning icon. On the right, a detailed table provides information about this blocked thread:

Thread Name	WebContainer : 1
State	Deadlock/Blocked
Monitor	Owns Monitor Lock on java/lang/Object@023EF870/023EF87C
	at com.ibm/issf/atjolin/badapp/BadAppServlet.dopeyMethod(BadAppServlet.java:320) at com.ibm/issf/atjolin/badapp/BadAppServlet.doPost(BadAppServlet.java:257) at javax/servlet/http/HttpServlet.service(HttpServlet.java:763) at javax/servlet/http/HttpServlet.service(HttpServlet.java:856) at com.ibm/ws/webcontainer/servlet/ServletWrapper.service(ServletWrapper.java:989) at com.ibm/ws/webcontainer/servlet/ServletWrapper.handleRequest(ServletWrapper.java:501)

© Copyright IBM Corporation 2013

Figure 6-31. Thread analysis: Monitor details

WA5913.0

Notes:

The Monitor Detail view provides a hierarchical tree of the threads.

Each top-level thread holds a lock. By clicking each thread to expand the tree, in the hierarchy you can see information about the threads that this thread is blocking.



IBM Whole-system Analysis of Idle Time (WAIT)

- New web-based tool from IBM Research
- WAIT report identifies bottlenecks such as:
 - Waiting for database data
 - Waiting on hot locks
 - Garbage collection degradation of performance
 - Inefficient code
- The report shows you the full stack context responsible for each bottleneck
- Access at <http://wait.ibm.com>
- Using WAIT
 - Generate javacore files on your system
 - Upload to WAIT-data is encrypted in transit
 - View report interactively in your browser

© Copyright IBM Corporation 2013

Figure 6-32. IBM Whole-system Analysis of Idle Time (WAIT)

WA5913.0

Notes:

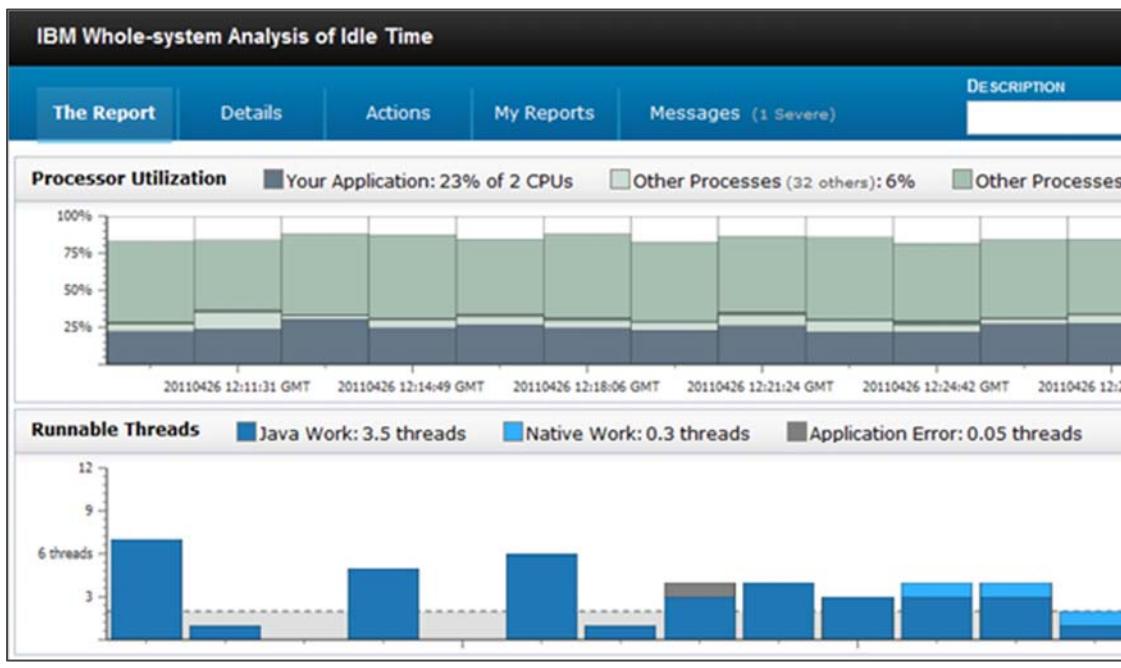
The **IBM Whole-system Analysis of Idle Time (WAIT)** tool helps you identify performance bottlenecks. It works with both stand-alone Java applications and distributed applications with a central Java tier.

Each WAIT report identifies bottlenecks such as: waiting for database data, waiting on hot locks, garbage collection activity, and inefficient code. The report shows you the full stack context responsible for each bottleneck.

WAIT is **zero-install**: it requires no monitoring agents, and the report is an interactive web page. You do not need to restart your application server, and the data collection time should be minimal. Infrequent snapshots usually suffice, and each snapshot pauses the JVM for usually under a second.

Example WAIT report

- The report can be viewed interactively by hovering the cursor over sections for more detail



© Copyright IBM Corporation 2013

Figure 6-33. Example WAIT report

WA5913.0

Notes:

The report has three timelines:

- The top (gray) timeline shows processor activity and how much your workloads contributed to it. Lots of dark gray is generally good meaning that your workloads were running smoothly for the entire duration over which data was collected.
- The middle (blue) timeline shows what Java work is running. This timeline generally mimics the top timeline in its form (although not always). It shows Java threads that are runnable (either running or capable of being run). Again, if there are high blue bars all the way across, it generally means that your workload was running unimpeded. However, there can still be hot code that contributes to poor performance. You can click the blue bars of the timeline or the aggregate summary at right to see the activities that use most of the processor. The middle timeline also shows when there is more work than the processors in the system can accommodate. This situation occurs whenever there is light blue above the dotted line (representing the number of processors in the system). Especially if there are such light blue caps uniformly over the course of execution, it suggests that the number of processors is underprovisioned, and that your workload's performance would increase with the addition of more processors.

3. The third timeline (not shown in the screen capture) shows Java threads that are delayed and not capable of being run at the current time. WAIT has a patented set of expert rules to diagnose these situations and the general reason why threads are delayed. In the example there are many such reasons: "Delayed by Remote Request", "Blocked on Monitor", "Delayed by Disk I/O", and "Waiting for GC Work" (GC = garbage collection). As with the middle timeline, you can click any of the individual bar segments or the legend at left to learn more about the causes of delay – more about that later. However, if your report shows any significant number of delayed threads, it suggests that the performance of your workload is subpar and can be improved with appropriate tuning of software or hardware. This situation is true even if the middle blue bars are exhausting the processor's capacity. Delays in the third timeline mean that work is taking longer to run and in transaction-oriented systems, likely increasing user response time, in addition to using resources like memory, thus reducing efficiency.

Unit summary

Having completed this unit, you should be able to:

- Describe what a hang is
- Detect a hang condition
- Trigger and analyze jvavacore files for hung threads
- Use the WebSphere Application Server hang detection facility
- Use the IBM Thread and Monitor Dump Analyzer for Java

© Copyright IBM Corporation 2013

Figure 6-34. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. True or false: The Thread Monitor facility is turned on by default for every application server.
2. True or false: A javacore file contains a dump of a JVM's threads at the time the file was generated.
3. True or false: An application can seem to be in a hung state if two threads are deadlocked.
4. True or false: The Thread and Monitor Dump Analyzer is part of the administrative console.

© Copyright IBM Corporation 2013

Figure 6-35. Checkpoint

WA5913.0

Notes:

Write your answers here:

1. True or false? Explain.
2. True or false? Explain.
3. True or false? Explain.
4. True or false? Explain.



Checkpoint answers

1. True
2. True
3. True
4. False: The Thread and Monitor Dump Analyzer runs in the IBM Support Assistant.

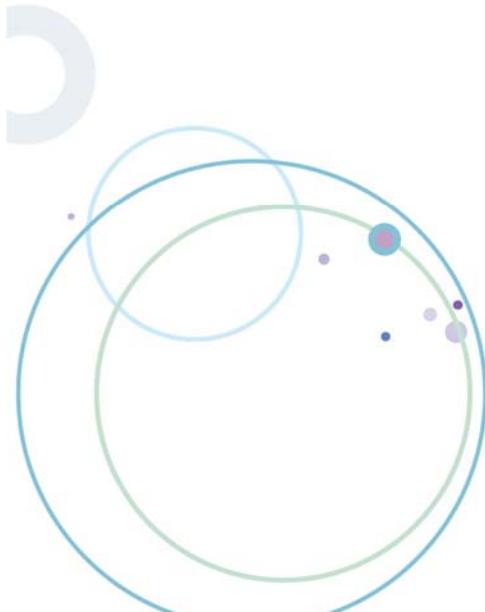
© Copyright IBM Corporation 2013

Figure 6-36. Checkpoint answers

WA5913.0

Notes:

Exercise 4



Troubleshooting hung threads

© Copyright IBM Corporation 2013

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

8.0

Figure 6-37. Exercise 4

WA5913.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Use the thread monitor hang detection facility
- Generate and analyze various hung thread scenarios
- Use wsadmin to trigger a javacore file
- Analyze the javacore file for hung threads by using the IBM Thread and Monitor Dump Analyzer for Java tool

© Copyright IBM Corporation 2013

Figure 6-38. Exercise objectives

WA5913.0

Notes:

Unit 7. How to troubleshoot crashes

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:

- Define what a crash is
- Detect a crash
- Analyze a javacore file for a crash
- Analyze system core files
- Describe the tools available for troubleshooting a crash
- Describe and use the IBM Monitoring and Diagnostic Tools for Java-Dump Analyzer tool

How you will check your progress

- Checkpoint questions
- Lab exercise

References

SG24-6798-00 WebSphere Application Server V6 Problem Determination
for Distributed Platforms

Unit objectives

After completing this unit, you should be able to:

- Define what a crash is
- Detect a crash
- Analyze a javacore file for a crash
- Analyze system core files
- Describe the tools available for troubleshooting a crash
- Describe and use the IBM Monitoring and Diagnostic Tools for Java - Interactive Diagnostic Data Explorer

© Copyright IBM Corporation 2013

Figure 7-1. Unit objectives

WA5913.0

Notes:



Topics

- Troubleshooting a crash
- Tools for troubleshooting crashes

© Copyright IBM Corporation 2013

Figure 7-2. Topics

WA5913.0

Notes:

7.1. Troubleshooting a crash

Troubleshooting a crash



© Copyright IBM Corporation 2013

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

8.0

Figure 7-3. Troubleshooting a crash

WA5913.0

Notes:

JVM process crash defined

- Not the same as thread hangs
- Symptoms
 - Process was stopped because of a Java exception or an OS signal
- Usual causes
 - Bad JNI call or library problem
 - Segmentation violations while running native code
 - Out-of-memory exception
 - Call stack overflow
 - Unexpected exception (for example, out of disk space)
 - Optimizer failure (for example, JIT)

© Copyright IBM Corporation 2013

Figure 7-4. JVM process crash defined

WA5913.0

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:

- Failures that are a result of some type of problem in the environment where the JVM runs, such as a lack of resources: OutOfMemory, StackOverflow, unexpected exception, and more. These problems are solved by fixing the application or the environment in which the problem occurred.
- Failures that are a result of an internal error (a defect) in the JVM implementation (for example, JIT). These problems are fixed by applying a JDK patch.
- Failures that are a result of an internal error in some external third-party JNI library that the customer uses and are loaded in the JVM. These problems are 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 produces a javacore file. With internal JVM errors and JNI errors, you often get a “hard” crash, with just a core file but no javacore file.

Crash problem determination: Data to collect

- Javacore files
 - Also known as jadump or thread dump files
 - Text file that an application server creates during a failure
 - Can also be triggered manually
 - Error condition is given at the top of the javacore file
- Core files
 - Also known as process dumps or system core files
 - Created by underlying operating system
 - Complete dump of the virtual memory for the process
 - Can be large
 - Tools available to parse files into readable formats
- Steps to collect data
 - Look for a javacore file that is automatically created during a crash
 - If no javacore was generated, look for a system core dump

© Copyright IBM Corporation 2013

Figure 7-5. Crash problem determination: Data to collect

WA5913.0

Notes:

When a process crashes, the underlying operating system often creates a 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 memory dump was taken can be extracted and analyzed.

The javacore files are text files that contain only a summary of the information that is most pertinent to the most common problems.

Javacores are enabled by default on IBM JDK; they can be disabled by setting the `DISABLE_JAVADUMP` environment variable to TRUE.

On Solaris and HP-UX, instead of a separate javacore file, a memory dump of all thread stacktrace 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: Output is written to a core file named `core.×tamp.&processid.dmp` that is in the current working directory, for example: `/opt/IBM/WebSphere/AppServer/profiles/Dmgr01`

Make sure that a full core dump is produced

- Underlying operating system might have settings that prevent creation of a full core dump when a process crashes
 - For example, the `ulimit` on UNIX systems might specify a limit on the size of core file that is too small and the core dump is truncated
 - Some operating systems have a parameter to control the type of core files
 - On AIX, use the command “`lsattr -Elsys0 | grep full`” to check whether `fullcore` is configured or not
- Make sure that a full core dump can be created before a problem occurs
 - Avoid recreation of problem, especially in a production environment

© Copyright IBM Corporation 2013

Figure 7-6. Make sure that a full core dump is produced

WA5913.0

Notes:

The procedure to enable full core dump is different depending on the operating system:

- For Linux: <http://www.ibm.com/support/docview.wss?&uid=swg21115658>
- For AIX: <http://www.ibm.com/support/docview.wss?&uid=swg21052642>
- For Windows: <http://www.ibm.com/support/docview.wss?&uid=swg21053924>

Diagnostics Collector

- You can configure the JVM to use the Diagnostics Collector to gather documentation and diagnostic data automatically after detecting a runtime problem such as a crash or out-of-memory condition.
 - Enable by using the generic JVM argument `-Xdiagnosticscollector`
- The Diagnostics Collector gathers system memory dumps, `javadoc` files, heap memory dumps, verbose GC log (if present), and JVM trace files that match the time stamp for the Java problem that caused the collector to start
 - Outputs a single compressed file
 - For example, `java.gpf.<time_stamp>.<event_ID>.<pid>.zip`
- Available on IBM JDK 1.6 and 1.7
- For more details see the topic “The Diagnostics Collector” in the Java Diagnostics Guide 6

© Copyright IBM Corporation 2013

Figure 7-7. Diagnostics Collector

WA5913.0

Notes:

The JVM automatically starts the Diagnostics Collector after detecting a runtime problem and producing diagnostic data. The Diagnostics Collector gathers diagnostic data files that the JVM produces and stores them in a single archive file.

If a system dump is produced, the Diagnostics Collector runs `jextract` on the memory dump and collects the `jextract` output. There is an option to disable this behavior if `jextract` is not required. All actions are logged in a text file, which is also copied into the output compressed file.

The Diagnostics Collector gathers system dumps, Java dumps, heap dumps, verbose GC log (if present), and JVM trace files that match the time stamp for the Java problem that caused the collector to start. The Diagnostics Collector copies the diagnostic files that it finds into a single output compressed file. The output file is named by using the type of problem that occurred, the time stamp of the problem event, and the process ID of the Java application, for example:
`java.gpf.20081219.105104.9221.zip`.

At JVM startup, the Diagnostics Collector runs a diagnostic configuration check. It checks for any settings that disable Java diagnostics. If any are found, a warning is printed on `stderr` and in the `JavaDiagnosticsCollector.log` file. Fix the settings that are identified from any warning messages before restarting your Java application. Fixing warnings makes it more likely that the

right data is available for IBM Support to diagnose a Java problem.

UNIX operating system common signals

- **SIGQUIT (kill -3)**
 - Indicates that a command was issued to generate a thread dump
 - Typically does not end the JVM process
- **SIGILL (kill -4)**
 - Means that an illegal instruction was issued
 - This signal 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 with JIT-compiled code
- **SIGSEGV (kill -11)**
 - Indicates an operation that is not valid in a program
 - Example: Accessing an illegal memory address
 - This signal is typically indicative of a programming problem in one of the native libraries

© Copyright IBM Corporation 2013

Figure 7-8. UNIX operating system common signals

WA5913.0

Notes:

Keep in mind that non-JVM processes (that is, non-Java-based applications) can 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 run 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 signal is the equivalent of a `kill -4` command. This signal means that an invalid instruction was run. This signal 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 that is a result of 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 invalid memory address. This signal is typically indicative of a programming problem in one of the native libraries.

Signals -1, 0, OUTOFMEMORY, and SIGNONE are the memory signals. These signals are described in a later unit.

Signals 10, 11, SIGBUS, and SIGSEGV indicate an application server crash.

Windows operating system common signals

- **Memory access error**
 - Invalid memory address
 - JVM action: javacore file and stop the process
- **Illegal access error**
 - JVM action: javacore file and stop the process

© Copyright IBM Corporation 2013

Figure 7-9. Windows operating system common signals

WA5913.0

Notes:

The signals that Windows uses are entirely different from the signals that UNIX uses. Here is a list of some of the operating system signals that are produced on Windows. Some signals also produce a user.dmp file. The JVM signal handler library is hpi.dll.

Javacore subcomponents helpful for crash debugging

TITLE	Shows basic information about the event that caused the generation of the javacore file, the time it was taken, and the file name
GPINFO	<ul style="list-style-type: none"> Shows some general information about the operating system If the memory dump resulted from 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 started the JVM process, and the JVM environment
THREADS	Identifies the current thread and provides a stack trace

© Copyright IBM Corporation 2013

Figure 7-10. Javacore subcomponents helpful for crash debugging

WA5913.0

Notes:

It is important to understand what signal caused the javacore. This signal 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 a “user” dump event)? If the javacore is written to diagnose a hung JVM process, the steps you take are different than if the javacore is due to a JVM crash.

Use the Java command line to determine whether 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** show the Java and native stacks of the current thread. The most recent calls are at the top of the stack.

- The Java stack shows the Java method calls that the current thread made.
- If there is a native stack, this native stack contains the most recent events that the thread ran. As the name implies, these events are in native code (libraries) called by Java.

This output can be used to determine what was running when the crash occurred.

To verify that the javacore file is complete, you should see an “END OF DUMP” string at the end of the file.

Javacore example that shows crash symptoms

```

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\javacore.20070925.152644.txt
0SECTION      GPINFO subcomponent dump routine
NULL
=====
2XHOSLEVEL    OS Level       : Windows XP 5.1 build 2600 Service Pa
2XHCPUS       Processors -
3XHCPUARCH    Architecture  : x86
3XHNUMCPUS    How Many      : 2
1XHEXCPMODULE Module: C:\WINDOWS\system32\msvcrt.dll ← C++ runtime library
1XHEXCPMODULE Module_base_address: 77C10000
1XHEXCPMODULE Offset_in_DLL: 000378C0
{deleted lines}
0SECTION      THREADS subcomponent dump routine
NULL
=====
1XMCURTHDINFO Current Thread Details
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)

```

© Copyright IBM Corporation 2013

Figure 7-11. Javacore example that shows crash symptoms

WA5913.0

Notes:

GPF stands for general protection fault.

For example, this failure can indicate that the javacore file was created as the result of a fault in the C++ runtime library (Microsoft Visual C++ runtime library). This output shows a problem that is related to code involving the Java Native Interface (JNI).



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 might be indicated, such as DB2

© Copyright IBM Corporation 2013

Figure 7-12. Javacore fault module

WA5913.0

Notes:

Look for the full path name to that library or module, as that can indicate the use of some third-party JNI library. It can also point to the use of some nonstandard library that is 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 whether you can use them to determine the library that caused the crash.

Sometimes, native code that the JIT compiler generates is the cause of the crash. One possible workaround for this type of crash is to determine the method on which the failure is occurring and have the JIT compiler skip this method. Upgrading the JDK to the latest level can also fix this problem.

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

```

© Copyright IBM Corporation 2013

Figure 7-13. Javacore current thread details (JDK 1.4.2)

WA5913.0

Notes:

If the library is not identified from the signal information, you must examine the current thread details. Look to see whether 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 libjitic.a library, which is generating native code for a Java method. This error 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 libjitic.a library.

Crashes during JIT compilation

- To see whether the JIT is failing in the middle of a compilation, use the `verbose` option with the following extra settings:
 - `-Xjit:verbose={compileStart|compileEnd}`
- Also check `native_stderr.log` to see whether it identifies 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 (for example, it starts compiling, but crashes before it can end)
 - Use the `exclude={methodname}` parameter to exclude it from compilation
 - For example, `-Xjit:exclude={java/lang/Math.max(II)I}`
 - If excluding the method prevents the crash, you have an excellent workaround that you can use while the service team corrects your problem

© Copyright IBM Corporation 2013

Figure 7-14. Crashes during JIT compilation

WA5913.0

Notes:

See the detailed description of this topic in the reference: IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 6 *Diagnostics Guide*, Chapter 27, JIT and AOT problem determination, Disabling the JIT or AOT compiler.

Steps if the cause of the crash is not identified

- Frequently, the javacore file does not clearly identify the cause of the signal
- Often the native stack shows the following message:

```
----- Native Stack -----  
unable to backtrace through native code - iar 0x3062e73c not  
in text area (sp is 0x2ff21748)
```

- Steps that you can take:

- 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 whether this provides more information
- Sometimes a bad Java SDK installation can cause problems

© Copyright IBM Corporation 2013

Figure 7-15. Steps if the cause of the crash is not identified

WA5913.0

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 that 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.

The problem might be in the JIT compiler. If so, you might be able to skip JIT compiling for a method, if the problem is in the code that is generated for a method or in the code generation itself.

Sometimes a bad Java SDK installation can cause problems. The full version for the javacore file comes from the `libjvm.a`. The full version for the `Java -fullversion` command is from the Java executable file. 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 whether a process dump (core file) exists. If it does, it is still likely that you have an application server crash.

7.2. Tools for troubleshooting crashes

Tools for troubleshooting crashes



© Copyright IBM Corporation 2013

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

8.0

Figure 7-16. Tools for troubleshooting crashes

WA5913.0

Notes:

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 memory dump
- Not just one tool: an extensible framework for building many different tools
- By providing common framework, the use of specific tools for specific JVM artifacts is avoided

© Copyright IBM Corporation 2013

Figure 7-17. What is DTFJ?

WA5913.0

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 framework allows tool writers to access information from the memory dump about the system, process, Java VM, and Java application without having to understand how the relevant structures are laid out in memory. This framework makes the ability to write postmortem tools far more accessible.

There are many different JRas artifacts, such as system dumps (core files, MiniDumps), heap dumps (.phd files), javacore files (javacore, .txt files), but no common tools. There are many good reasons for different file formats, but often tools are JVM software-specific (jformat, jcore, kca).

The DTFJ-based Dump Analyzer tool was created to convert system dump files into human readable format.

In the past, there were various other ad hoc tools, and sometimes people even used a simple low-level debugger (for example, dbx, gdb) to try to understand a crash. It is not as good as running a DTFJ tool, but sometimes that is all you can do.



Components of the DTFJ family

- **jextract**: A tool to capture information from a JVM system memory dump (for example, core file) and package it into a platform-independent format
- DTFJ library or core library: A library that parses the contents of the system dump file that jextract packages, 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

© Copyright IBM Corporation 2013

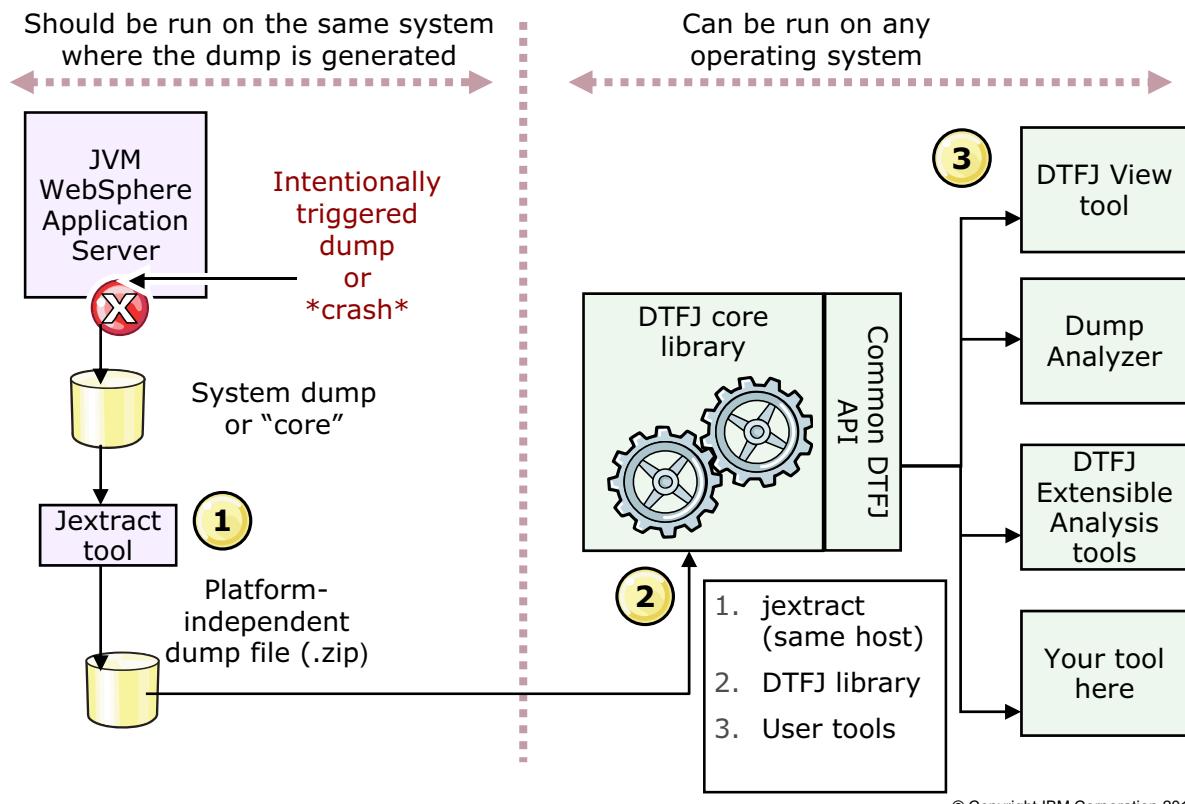
Figure 7-18. Components of the DTFJ family

WA5913.0

Notes:

The jextract tool is in <was_root>\java\jre\bin.

An example of using the DTFJ components



© Copyright IBM Corporation 2013

Figure 7-19. An example of using the DTFJ components

WA5913.0

Notes:

IBM support requires that the *jextract* tool run on the same machine where the memory dump was generated.

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 that is called *jextract*, which processes a core file and produces a platform-independent file suitable for the main phase of analysis
- A library, the DTFJ dump reader library, which does the bulk of the analysis
- A program that is called a *Dump Analyzer*, which drives the library to extract particular pieces of information for processing

Where is DTFJ supported?

- jextract and the main DTFJ runtime library are now provided and the standard IBM JDK supports them
 - IBM JDK 1.4.2 SR4 and beyond: WebSphere Application Server 5.1, 6.0
 - IBM JDK 1.4.2 SR4 for 64-bit systems: WebSphere Application Server 6.0.2
 - IBM JDK 1.5.0 SR1 and beyond: WebSphere Application Server 6.1
 - IBM JDK 1.6.0
 - All IBM JDK systems: AIX, Linux, Windows, z/OS, iSeries, 32-bit, and 64-bit
- Tools must be obtained separately within IBM Support Assistant
- You might 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 is able to extract from that dump
- Not currently supported for non-IBM JDKs such as Solaris and HP-UX

© Copyright IBM Corporation 2013

Figure 7-20. Where is DTFJ supported?

WA5913.0

Notes:

To emphasize, DTFJ supports only the IBM JDKs beginning with JDK 1.4.2 SR4. These JDKs include IBM JDK 5, which WebSphere Application Server V6.1 uses, and JDK 6 in WebSphere 7.0 and 8.0.

The open source project Kato aims to make DTFJ technology available to non-IBM JDK:
<http://incubator.apache.org/kato/site/index.html>

What is the Interactive Diagnostic Data Explorer?

- Interactive Diagnostic Data Explorer (IDDE) is a GUI-based alternative to the dump viewer (`jdumpview` command)
- IDDE provides the same function as the dump viewer, but with extra support such as the ability to save command output
- Use IDDE to more easily explore and examine dump files that the JVM produces
- Within IDDE, you enter commands in an investigation log to explore the dump file
- The support that is available in the investigation log includes the following items:
 - Command assistance
 - Auto-completion of text, and some parameters such as class names
 - The ability to save commands and output, which you can then send to other people
 - Highlighted text and flagging of issues
 - The ability to add your own comments
 - Support for using the Memory Analyzer from within IDDE

© Copyright IBM Corporation 2013

Figure 7-21. What is the Interactive Diagnostic Data Explorer?

WA5913.0

Notes:

Interactive Diagnostic Data Explorer (IDDE) is the strategic tool for allowing interactive analysis of JVM problems with post mortem artifacts such as core files or javacores. It is a lightweight tool that allows you to quickly get information from the artifact you are investigating where you are not sure what the problem is and you want to avoid starting resource-intensive analysis.



Using Interactive Diagnostic Data Explorer (IDDE)

- IDDE supersedes the Dump Analyzer tool
- Distributed and accessed through IBM Support Assistant
- Based on DTFJ, which provides cross system support
- Supports
 - Java core files
 - heap dumps (PHD file format)
 - system dumps (also known as core files)
 - Also supports multiple dump files that are contained in a compressed file, and dumps that were created on the z/OS operating system

© Copyright IBM Corporation 2013

Figure 7-22. Using Interactive Diagnostic Data Explorer (IDDE)

WA5913.0

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.ibm.com/developerworks/java/jdk/diagnosis/>

IDDE features

Attempts to diagnose common JVM problems

- Deadlock in Java code
 - Report thread names, locations, and other information
- Out-of-memory condition
 - Report populations and large collections, and large objects
 - Summarize the native memory usage
- Internal error (general protection fault, segmentation violation)
 - Is failure in non-IBM native code?
 - Probably use coding error, report location, and other details
 - If using JDK V5 or later, it might suggest running with `-Xcheck:jni` command-line option
 - Otherwise, call IBM Support
- Otherwise, it generates a default summary report
 - Suggested action is to call IBM Support and provide the output

© Copyright IBM Corporation 2013

Figure 7-23. IDDE features

WA5913.0

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. The Dump Analyzer can be 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 `-Xcheck:jni` command-line option causes the JVM to monitor JNI usage. When `-Xcheck:jni` 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, with the JNI functions available.

IDDE: jvm.DumpReport contents

- Basic information about the JVM process
 - Processor type, process ID, command line, JVM version, and other information
- JVM initialization arguments
 - System class path, heap tuning parameters, and other command-line options
- Environment variables
- Native libraries that are loaded in this process
- Threads (both Java threads and native threads)
 - Java thread ID, WebSphere Application Server thread ID, `java.lang.Thread` object, priority, and other thread details
 - Java stack, native stack
- Monitors and locks
- Heap memory layout

© Copyright IBM Corporation 2013

Figure 7-24. IDDE: jvm.DumpReport contents

WA5913.0

Notes:

This version of DefaultDumpReport is the standard version, which includes the most commonly used information:

- JVM initialization arguments
- Lists of threads and thread stacks with detailed information that includes:
 - Monitors held or waited for by each thread
 - WebSphere Application Server thread IDs (if applicable)
- Detailed information about loaded libraries
- Basic information about the Java heaps
- Environment variables
- List of Java monitors, cross-referenced against the threads that hold these monitors

WebSphere Education

IDDE: Dump report

The screenshot shows the IDDE interface with the following details:

- Launch Activity:** Analyze Problem
- IBM Monitoring and Diagnostic Tools...**
- PD Navigator:**
 - IDDE_project
 - Linked sources
 - /opt/IBM/WebSphere/AppServer/... (Type: core)
 - Current Investigation
 - Investigation Log
 - 0x0 - JRE 1.6.0 20130301 (selected)
- Outline:**
 - Ihelp
 - Ibasicinfo
 - Idumpreport (selected)
 - Ideadlock
- Main Content Area:**

```
!dumpreport {
=====
1. Results from Analyzer=com.ibm.dtfj.analyzer.jvm.DumpReport
=====

Analyzer full name: com.ibm.dtfj.analyzer.jvm.DumpReport
Analyzer version: 1.3.0.20070812
Analyzer description:
Report basic information from this JVM image (similar to javacore) - Standard version

1.1. ===== Image and runtime information =====

Now reporting on runtime: 0.0.0

Image: (no identity)
Time of dump: [<unavailable>]
System Type: Linux
Processor Type: x86
Number of Processors: 1
Installed Memory: 3088556032
This Image contains: 1 address spaces; 1 processes; 1 runtimes

Process: PID:31811
```

© Copyright IBM Corporation 2013

Figure 7-25. IDDE: dump report

WA5913.0

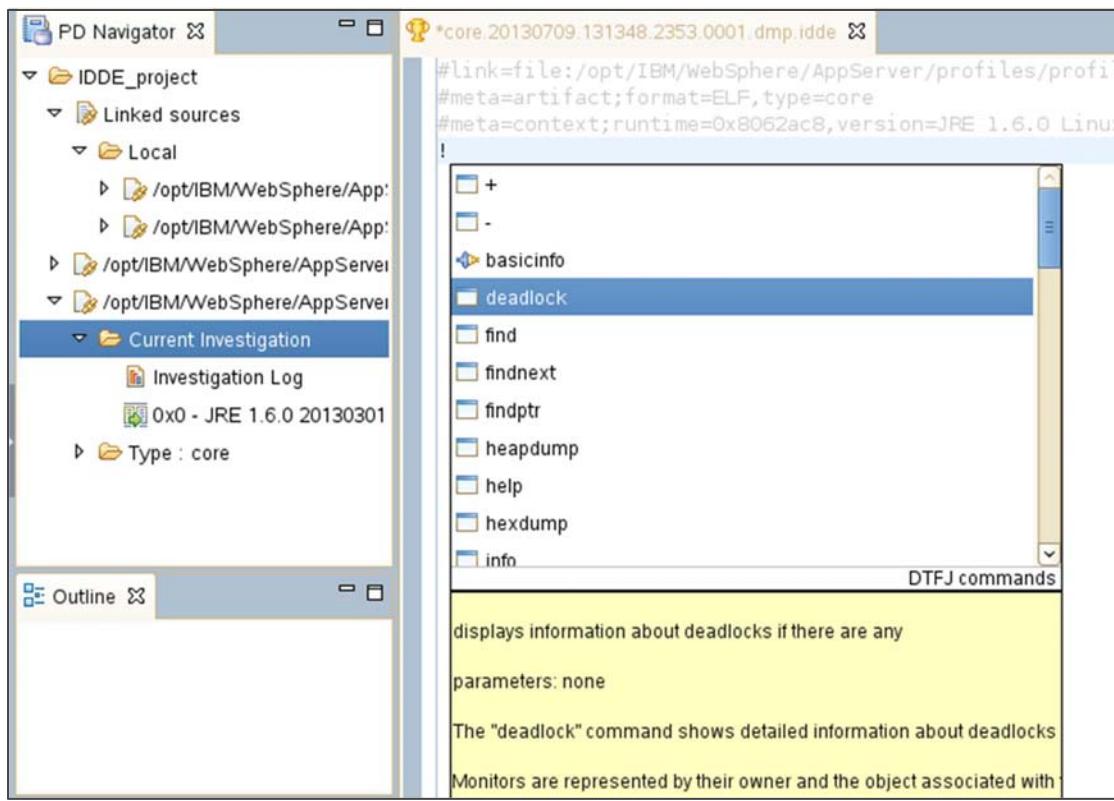
Notes:

After starting IDDE, follow the instructions to create a new IDDE project:

1. Right-click the IDDE Project, and select New PD Artifact.
2. Go to the dump artifact and choose “Link to artifact”.
3. Now, expand the linked artifact, and expand “Start Investigation”.
4. This action opens a text editor for the memory dump. You can make notes and save in the editor to track your investigation.
5. Type ! in any white space to list available IDDE commands. Type a second ! to list available IBM Extensions for Memory Analyzer commands. Finally, type a third ! to list available Dump Analyzer commands. Select the !!!dumpreport command, and type Ctrl+Enter to run it.



IDDE: Help menu for DTFJ commands



© Copyright IBM Corporation 2013

Figure 7-26. IDDE: Help menu for DTFJ commands

WA5913.0

Notes:

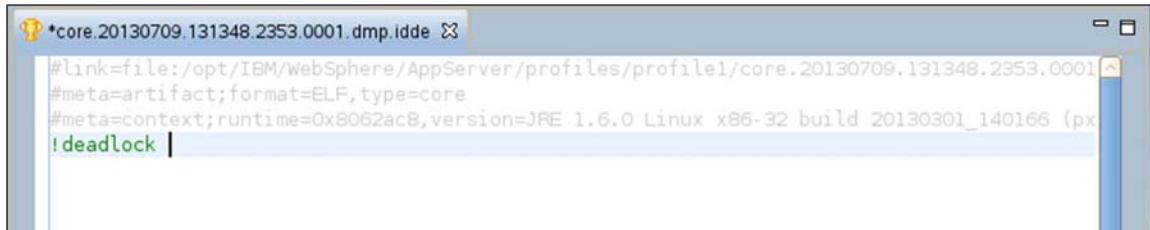
To enter a command, start by typing an exclamation mark (!). The command assistance function displays a list of available commands.

Use the command list to help you choose a command to run. Click a command in the list to see information about that command. Click the information window to display scroll bars, if required.

The label in the bar at the bottom of the list window shows the type of commands in the list. You might have multiple lists; the default list is DTFJ commands, which contain the jdumpview commands and, if the dump file contains a JVM, other commands that are specific to IDDE. The IDDE commands are marked with a plug-in icon. Other lists might also be available. For example, if you installed IBM Monitoring and Diagnostic Tools for Java – Dump Analyzer, you can view a separate list for the Dump Analyzer commands. To navigate between lists, click the bar at the bottom of the list window, press Ctrl+Space, or type another exclamation mark.

IDDE: Deadlocked threads detection

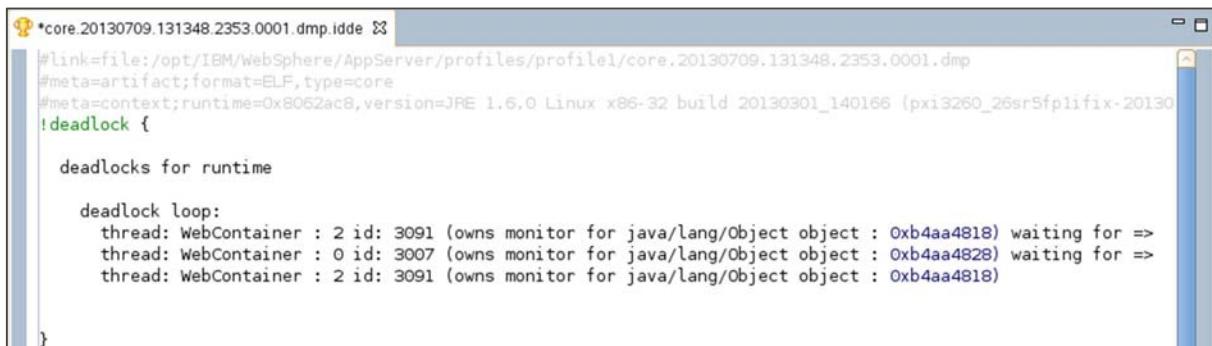
- Type !deadlock (or select from the command assistance menu)



The screenshot shows the IDDE (IBM Diagnostic Data Editor) interface. A terminal window is open with the command line visible. The command entered is !deadlock. The terminal output shows basic dump file metadata followed by the command itself.

```
*core.20130709.131348.2353.0001.dmp.idde
#link=file:/opt/IBM/WebSphere/AppServer/profiles/profile1/core.20130709.131348.2353.0001
#meta=artifact;format=ELF,type=core
#meta=context;runtime=0x8062ac8,version=JRE 1.6.0 Linux x86-32 build 20130301_140166 (pxi3260_26sr5fp1ifix-20130301_140166)
!deadlock |
```

- Type Ctrl+Enter to run the command



The screenshot shows the IDDE interface after running the !deadlock command. The output indicates a deadlock loop involving three threads, all identified as WebContainer threads. Thread 2 is waiting for a lock held by Thread 0, and Thread 0 is waiting for a lock held by Thread 2.

```
*core.20130709.131348.2353.0001.dmp.idde
#link=file:/opt/IBM/WebSphere/AppServer/profiles/profile1/core.20130709.131348.2353.0001.dmp
#meta=artifact;format=ELF,type=core
#meta=context;runtime=0x8062ac8,version=JRE 1.6.0 Linux x86-32 build 20130301_140166 (pxi3260_26sr5fp1ifix-20130301_140166)
!deadlock |

deadlocks for runtime

deadlock loop:
    thread: WebContainer : 2 id: 3091 (owns monitor for java/lang/Object object : 0xb4aa4818) waiting for =>
    thread: WebContainer : 0 id: 3007 (owns monitor for java/lang/Object object : 0xb4aa4828) waiting for =>
    thread: WebContainer : 2 id: 3091 (owns monitor for java/lang/Object object : 0xb4aa4818)

}
```

© Copyright IBM Corporation 2013

Figure 7-27. IDDE: Deadlocked threads detection

WA5913.0

Notes:

The `deadlock` command detects deadlock situations in the Java application that was running when the system dump was produced. Example output is shown on this slide.

Threads are identified from their Java thread name, whereas object monitors are identified from the address of the object in the Java heap. You can use the `x/J <0xaddr>` command to obtain further information about the threads that use the `info thread *` command. You can obtain further information about the monitors. In this example, the deadlock analysis shows that Thread-2 is waiting for a lock that is held by Thread-0, which in turn waits for a lock that is held earlier by Thread-2.

IDDE: JIT compiled methods

- Use the `info jitm` command to list compiled threads

```
#core.20130709.145506.17546.0001.dmp.idde
#meta=artifact;format=ELF,type=core
#meta=context;runtime=0x8062ae8,version=JRE 1.6.0 Linux x86-32 build 20130301_140166 (pxi3260_26sr5fp1if)
!info jitm {}

Info jitm
displays JIT'ed methods and their addresses
parameters: none
prints the following information about each JIT'ed method
- method name and signature
- method start address
- method end address

start=0x980edb00 end=0x980ee837 javax/el/BeanELResolver::property(Ljavax/el/ELContext;Ljava/lang/C
start=0x980ea0c0 end=0x980ea232 javax/el/ArrayELResolver::getValue(Ljavax/el/ELContext;Ljava/lang/
start=0x9808f6e0 end=0x98090020 javax/el/ListELResolver::getValue(Ljavax/el/ELContext;Ljava/lang/C
start=0x9808e4e0 end=0x9808e6d7 javax/el/ResourceBundleELResolver::getValue(Ljavax/el/ELContext;Lj
start=0x9808f560 end=0x9808f6ba javax/el/MapELResolver::getValue(Ljavax/el/ELContext;Ljava/lang/O
start=0x98021240 end=0x98021265 javax/el/ELContext::<init>()
start=0x9808be00 end=0x9808bf94 javax/el/ELContext::getContext(Ljava/lang/Class;)Ljava/lang/Object
start=0x99caaa00 end=0x99caa1f javax/el/ELContext::setPropertyResolved(Z)V
start=0x99ca9e80 end=0x99ca9e9b javax/el/ELContext::isPropertyResolved()Z
start=0x980ddb80 end=0x980defa2 com/ibm/websphere/samples/pbw/jpa/Inventory::pcReplaceField(I)V
start=0x99d33da0 end=0x99d33e00 javassist/bytocode/Bytecode::add(I)V
```

© Copyright IBM Corporation 2013

Figure 7-28. IDDE: JIT compiled methods

WA5913.0

Notes:

The `info jitm` command displays JIT compiled methods and their addresses:

- Method name and signature
- Method start address
- Method end address

For a complete list of commands, see the topic “Commands available in jdumpview” in the Java 6 Information Center.

<http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp?topic=%2Fcom.ibm.java.doc.diagnostics.60%2Fhomepage%2Fplugin-homepage-java6.html>

Unit summary

Having completed this unit, you should be able to:

- Define what a crash is
- Detect a crash
- Analyze a javacore file for a crash
- Analyze system core files
- Describe the tools available for troubleshooting a crash
- Describe and use the IBM Monitoring and Diagnostic Tools for Java - Interactive Diagnostic Data Explorer

© Copyright IBM Corporation 2013

Figure 7-29. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. True or false: A common reason application servers crash is due to the problems with the JIT compiler.
2. True or false: A javacore file is always generated when an application server crashes.
3. True or false: System core files are in text format and can be analyzed manually by using a text editor.

© Copyright IBM Corporation 2013

Figure 7-30. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

1. True or false? Explain.

2. True or false? Explain.

3. True or false? Explain.

Checkpoint answers

1. True
2. False: Sometimes only a system core file is generated.
3. False: System core files are generated in binary format.

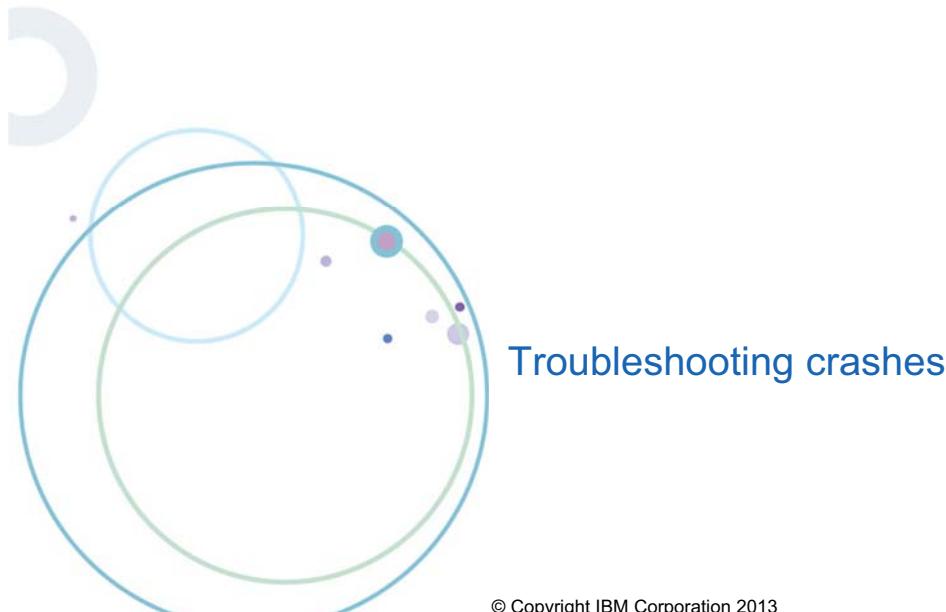
© Copyright IBM Corporation 2013

Figure 7-31. Checkpoint answers

WA5913.0

Notes:

Exercise 5



© Copyright IBM Corporation 2013

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

8.0

Figure 7-32. Exercise 5

WA5913.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Analyze a javacore file for a crash
- Use various methods to trigger a system core dump
- Use the IBM Monitoring and Diagnostic Tools for Java - Interactive Diagnostic Data Explorer to analyze system core files

© Copyright IBM Corporation 2013

Figure 7-33. Exercise objectives

WA5913.0

Notes:

Unit 8. Introduction to WebSphere out-of-memory problems

What this unit is about

This unit describes how to troubleshoot out-of-memory conditions.

What you should be able to do

After completing this unit, you should be able to:

- Define out-of-memory conditions
- Use monitoring tools to detect out-of-memory conditions
- Obtain and interpret a verbose GC log by using GCMV
- Obtain and analyze heap dumps and system core dumps
- Describe tools for analyzing out-of-memory problems

How you will check your progress

- Checkpoint
- Lab exercises

References

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:

www.ibm.com/support/docview.wss?uid=swg27013400&aid=1

Unit objectives

After completing this unit, you should be able to:

- Define out-of-memory conditions
- Use monitoring tools to detect out-of-memory conditions
- Obtain and interpret a verbose GC log by using GCMV
- Obtain and analyze heap dumps and system core dumps
- Describe tools for analyzing out-of-memory problems

© Copyright IBM Corporation 2013

Figure 8-1. Unit objectives

WA5913.0

Notes:



Topics

- OutOfMemory error overview
- Interpret verbose GC output
- Analyze Java heap dumps and system core dumps

© Copyright IBM Corporation 2013

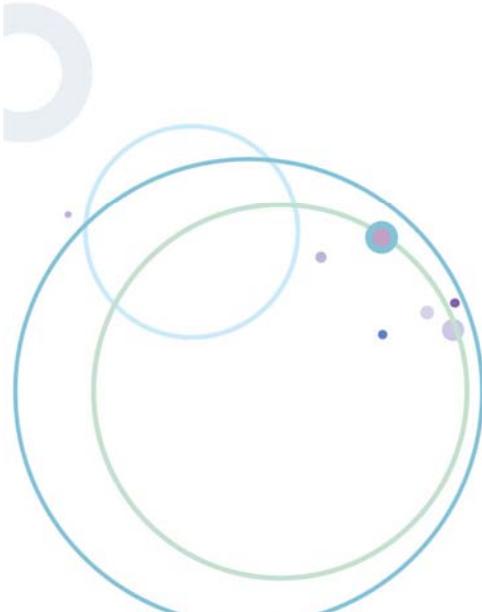
Figure 8-2. Topics

WA5913.0

Notes:

8.1. OutOfMemory error overview

OutOfMemory error overview



© Copyright IBM Corporation 2013

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

8.0

Figure 8-3. OutOfMemory error overview

WA5913.0

Notes:

What is a java.lang.OutOfMemoryError?

- Java virtual machine error
- Not enough memory to allocate an object; some of the causes are:
 - The Java heap is too small
 - Memory is available in the heap, but it is fragmented (rare for Java 6)
 - 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 as the cause for the javacore
 - Check heap size information
- Generate and analyze garbage collection and memory information:
 - Verbose GC
 - Javacore
 - Heap dump
 - System dumps

© Copyright IBM Corporation 2013

Figure 8-4. What is a java.lang.OutOfMemoryError?

WA5913.0

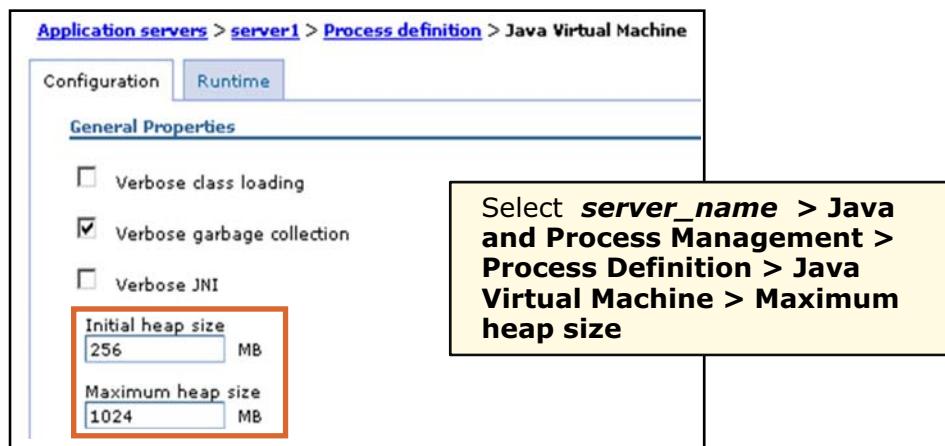
Notes:

An OutOfMemoryError (OOM) is a Java exception that occurs when the JVM is unable to allocate memory to satisfy a request. There are four potential reasons for an OOM to occur. The first reason is that the Java heap is too small for the amount of workload. Second is that the memory in the Java heap is fragmented and there is not enough contiguous free space to satisfy the request. Third is that some part of the Java code is leaking memory and the heap space is exhausted. Finally, the JVM might not have enough native memory to call operations that are associated with the Java object allocation. In any case, the first artifact to analyze is the javacore that is created by default whenever an OOM occurs; however, verboseGC logs and heap dumps are also valuable in resolving OOMs.



JVM heap is too small

- How do you know that the heap is too small?
 - If the JVM heap shows constant growth until it reaches the maximum heap size, it is too small
 - The JVM heap never achieves a steady state if it is too small
- Must increase the Java maximum heap size in the administrative console



© Copyright IBM Corporation 2013

Figure 8-5. JVM heap is too small

WA5913.0

Notes:

The first probable cause for an OOM is that the heap is not large enough for the workload. This problem often surfaces shortly after the JVM begins to process the workload, and the heap usage increases until the maximum available space is used. A steady increase in heap use can also be an indicator of a memory leak; you are going to explore the differences when you get to the topic of memory leaks. For now, if you suspect the heap is too small, the size can be increased under the Java virtual machine section of the server's Process Definition area of the administrative console.

Memory fragmentation

- Space is available on the heap, but not in contiguous blocks large enough to allocate most objects
- Some fragmentation always occurs
 - Should be treated as if the heap were too small
 - Tenured size might be too small for generational garbage collection
- The object that is being allocated is excessively large
 - The JVM is required 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 that is being allocated
 - If that is not possible, increase the JVM heap
- Heap fragmentation is not common in recent versions of Java (5 and later) because of improved garbage collection modes (gencon and balanced)

© Copyright IBM Corporation 2013

Figure 8-6. Memory fragmentation

WA5913.0

Notes:

The second potential reason for an OOM is memory fragmentation. This category was much more common for Java 1.4.2 and earlier. Therefore, it is unlikely to be the reason for an OOM in WebSphere 8.0; however, it is not impossible. Java objects must be allocated into a contiguous space of free memory; therefore it is possible that total free memory is large enough to satisfy the object, but it is not in a contiguous block. If the allocation size is relatively large (over 1 MB is a good indicator of large), then the best solution might be for developers to redesign the application to no longer require such large objects. If that is not possible, then increasing the overall heap size or tenured size for generational collection is the next step.

Symptoms of memory leak in the Java code

- No matter what the JVM maximum heap size is set to, the heap is still going to run out of memory
- Increasing the maximum heap size merely 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
 - **“Iceberg objects”**: Threads, classes, and direct bytebuffers that have a small Java heap footprint, but a large native heap footprint

© Copyright IBM Corporation 2013

Figure 8-7. Symptoms of memory leak in the Java code

WA5913.0

Notes:

The third and most common cause for an OOM is a leak in Java memory usage. This problem occurs when some part of the code is allocating objects without ever releasing them, and therefore the heap grows until objects no longer fit. Unlike when the maximum heap size is too small, increasing the heap size when there is a memory leak prolongs the inevitable OOM. Increasing the heap can make matters worse because more data can be lost when the OOM occurs. Memory use that grows over time is called a memory leak. While memory leaks can occur quickly, they often take an extended time to reach maximum heap. Most memory leaks occur because of large collections of objects; however, it is also possible that a few large objects are occupying most of the heap. Memory leaks can occur outside of the Java heap in native memory.

Class loader memory leaks

- Many memory leaks manifest themselves as class loader leaks
 - Classes with the same name can be loaded multiple times in a single JVM, each in a different class loader
 - The class loader retains a reference to every class it loaded
 - These references often remain after a web application reload
 - With each reload, more classes are pinned which leads to an out of memory error
 - The application code or JRE triggered code can cause class loader memory leaks
 - Beginning in V8.5, you can configure WebSphere Application Server to detect, prevent, and take action, if possible, on class loader memory leaks by using the memory leak detection policy
- You can also use the IBM Classloader Analyzer tool, available in IBM Support Assistant, to detect and analyze class loader problems

© Copyright IBM Corporation 2013

Figure 8-8. Classloader memory leaks

WA5913.0

Notes:

Many memory leaks manifest themselves as Classloader leaks. The name and Classloader uniquely identify a Java class. Classes with the same name can be loaded multiple times in a single JVM, each in a different Classloader. Each web application gets its own Classloader, and this Classloader is what WebSphere Application Server uses for isolating applications.

An object retains a reference to the class it is an instance of. A class retains a reference to the Classloader that loaded it. The Classloader retains a reference to every class it loaded. Retaining a reference to a single object from a web application pins every class that the web application loads. These references often remain after a web application reload. With each reload, more classes are pinned, which leads to an out-of-memory error.

The application code or JRE-triggered code typically causes Classloader memory leaks.

See the information center topic, “Configuring the memory leak policy”:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.nd.doc%2Fae%2Fttrb_configmemleak.html

To find more information on the IBM Classloader Analyzer, see the developerWorks article, “What is the IBM Classloader Analyzer?”:

[https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/
communityview?communityUuid=a0a94b0d-38fe-4f4e-b2e6-4504b9d3f596](https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=a0a94b0d-38fe-4f4e-b2e6-4504b9d3f596)

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
- The JVM attempts to create an OutOfMemoryError, but might not have the necessary resources

© Copyright IBM Corporation 2013

Figure 8-9. Not enough native memory

WA5913.0

Notes:

The final cause for an OOM is when there is not enough native memory to satisfy an object allocation. There is rarely a correlation between the size of the Java object allocation request and the amount of space that is required in native memory. Unfortunately the verbose GC logs do not help identify the cause of a native memory problem. In addition, the JVM tries to create an OOM but at times might not have the resources to even create that object.

Java process restrictions

- Not all Java process space is available to the Java application heap
- The Java runtime needs memory for:
 - The Java virtual machine resources
 - Backing resources for some Java objects
- This memory area is part of the native heap
- Most memory that is not allocated to the Java heap is available to the native heap
 - Available memory space minus Java heap \approx native heap
- Effectively, the Java process maintains two memory pools: Java and native

© Copyright IBM Corporation 2013

Figure 8-10. Java process restrictions

WA5913.0

Notes:

Java process spaces can be logically broken into two key memory segments: native memory and Java heap memory. The native memory is where the JVM allocates its own resources and interacts with the underlying OS. Some Java objects have native memory requirements (such as Type 2 JDBC drivers and I/O buffers), but the size of the native requirement is often not associated with the size of the Java object. The Java process space also contains the native libraries. An easy way to approximate the size of the native memory available to the process is to subtract the maximum Java heap size from the total memory available to the process space. This approximation is not accurate for Windows because of how native libraries are allocated.

The native heap

- Allocated using `malloc()` and `mmap()` 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 cannot directly be controlled
 - To increase this memory space, decrease the Java heap size

© Copyright IBM Corporation 2013

Figure 8-11. The native heap

WA5913.0

Notes:

The `malloc()` and `mmap()` methods are used to allocate memory in the native heap; therefore it is subject to management by the operating system. There are several reasons why a Java process must allocate space in the native memory, and the best way to determine native memory needs is to monitor the Java process from the operating system. The size of native memory needs cannot be directly controlled; however, decreasing the maximum Java heap effectively increases the space available for native memory.

Using Tivoli Performance Viewer to anticipate an OutOfMemoryError

- Tivoli Performance Viewer runs in the administrative console
- Uses Performance Monitoring Infrastructure (PMI) to capture information about the WebSphere run time
- Provides graphs and charts of the captured PMI data

© Copyright IBM Corporation 2013

Figure 8-12. Using Tivoli Performance Viewer to anticipate an OutOfMemoryError

WA5913.0

Notes:

One of the built-in tools for monitoring WebSphere for an OOM is Tivoli Performance Viewer. Tivoli Performance Viewer uses the Performance Monitoring Infrastructure (PMI) to display information about the WebSphere runtime and provides a graphical display of the information. Tivoli Performance Viewer runs on the Deployment Manager; however, it must communicate with the Application Server (often through the Node Agent) and can have a performance impact if used during a heavy load.



Administrative console PMI settings

Performance Monitoring Infrastructure (PMI)

Select:
Performance and Tuning > PMI > server_name

Runtime Configuration

General Properties

Enable Performance Monitoring Infrastructure

Use sequential counter updates

Currently monitored statistic set

None
No statistics are enabled.

Basic
 Provides basic monitoring, including Java EE and the top 38 statistics.
 JVM Runtime.HeapSize
 JVM Runtime.UsedMemory
 JVM Runtime.UpTime
 JVM Runtime.ProcessCpuUsage
 JCA Connection Pools.CreateCount
 JCA Connection Pools.CloseCount

Extended
 Provides extended monitoring, including the basic level of monitoring plus workload monitor, performance advisor, and Tivoli resource models.
 JDBC Connection Pools.JDBCTime
 JVM Runtime.HeapSize
 JVM Runtime.FreeMemory
 JVM Runtime.UsedMemory
 JVM Runtime.UpTime
 JVM Runtime.ProcessCpuUsage

© Copyright IBM Corporation 2013

Figure 8-13. Administrative console PMI settings

WA5913.0

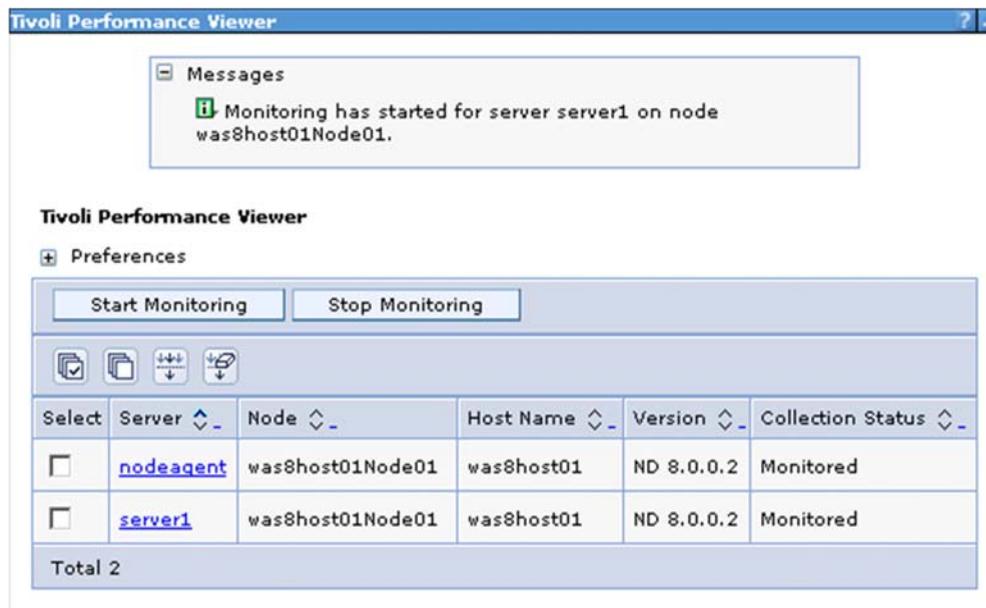
Notes:

PMI must be enabled and metrics must be collected before Tivoli Performance Viewer can be used on an Application Server. Here is a screen capture of where the basic metrics are being collected for server1.



Administrative console Tivoli Performance Viewer monitoring

- Select Monitoring and Tuning > Performance Viewer > Current Activity > *server_name*, and click Start Monitoring



© Copyright IBM Corporation 2013

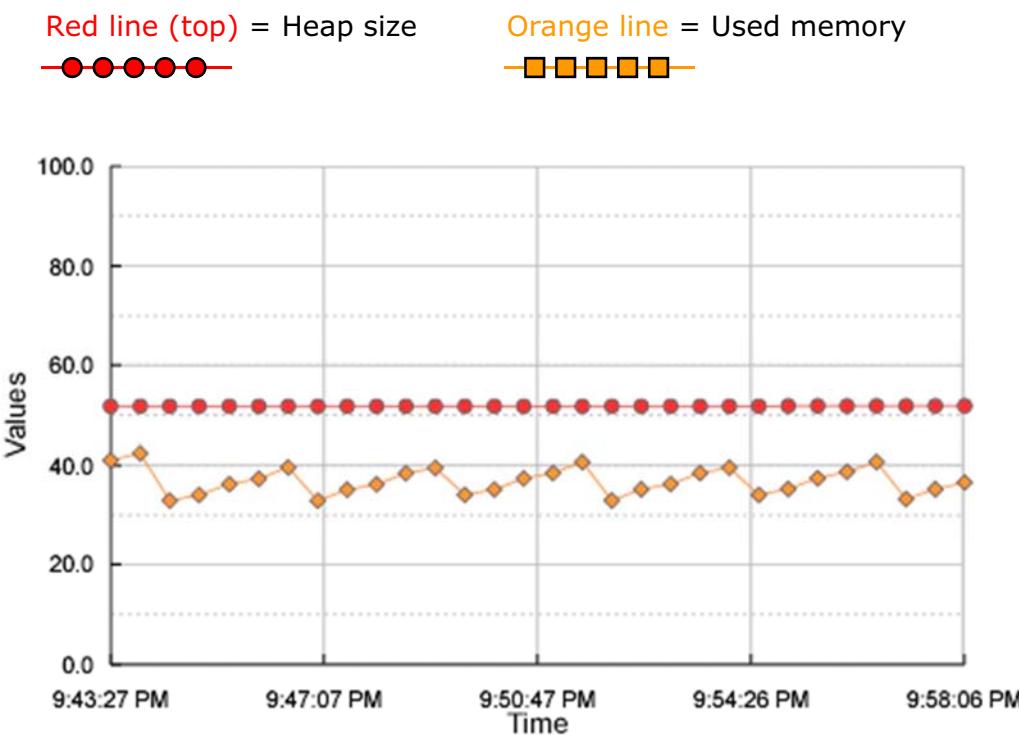
Figure 8-14. Administrative console Tivoli Performance Viewer monitoring

WA5913.0

Notes:

PMI is enabled and the metrics are collected. An administrator can use Tivoli Performance Viewer to start monitoring the Application Server.

Administrative console Tivoli Performance Viewer graph



© Copyright IBM Corporation 2013

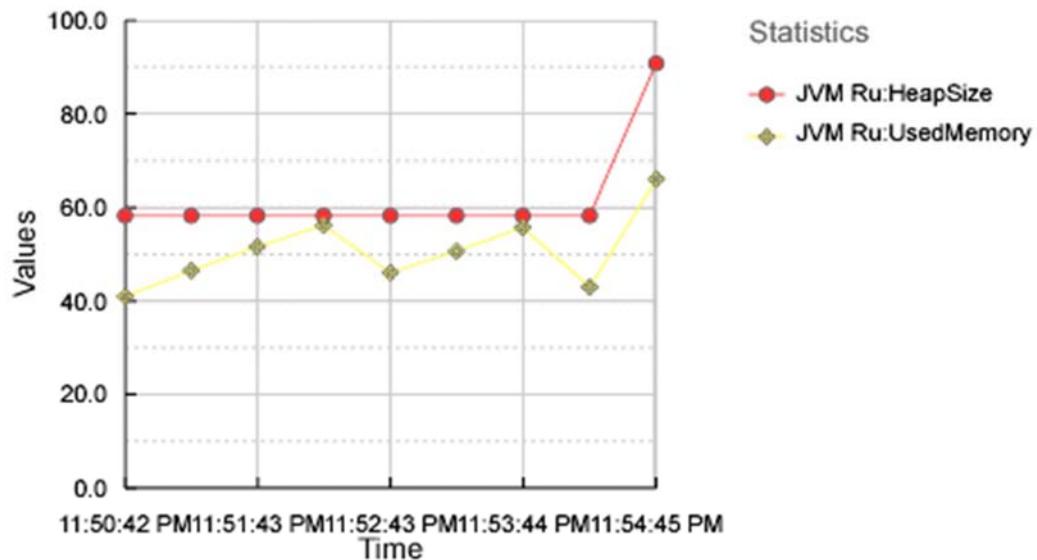
Figure 8-15. Administrative console Tivoli Performance Viewer graph

WA5913.0

Notes:

This screen is a Tivoli Performance Viewer graph of heap size versus used memory. The sawtooth pattern of the used memory line is indicative of a correctly working JVM. If the line steadily approaches the heap size (red line), it indicates either a leak or a need for a larger heap size.

Tivoli Performance Viewer graph: Heap expansion



© Copyright IBM Corporation 2013

Figure 8-16. Tivoli Performance Viewer graph: Heap expansion

WA5913.0

Notes:

Observe that the used heap grows to heap size after the first two garbage collections. Following the garbage collection, the heap expanded. A sudden burst of application activity, requests for large objects, or possibly a memory leak can cause the heap to grow rapidly.

Tivoli Performance Viewer usage

- Tivoli Performance Viewer
 - Monitor internal thread pools and heap statistics
- Know the expected behavior of your application: I/O intensive, JMS, number of EJBs, expected load requirements
- Use Tivoli Performance Viewer to monitor the heap usage
 - If the used heap continues to grow overtime with no corresponding increase in user load, there might be a memory leak
 - Use a profiler to dig into the specific heap to determine where the problem exists
 - JVMTI is a native programming interface that provides tools to inspect the state of the Java virtual machine (JVM)

© Copyright IBM Corporation 2013

Figure 8-17. Tivoli Performance Viewer usage

WA5913.0

Notes:

Tivoli Performance Viewer can be used to monitor several aspects of both an application and the WebSphere Application Server itself. The information in Tivoli Performance Viewer is best used when combined with existing expectations for how those metrics should behave in production.

JVMTI collects information about the JVM that runs the application server. The Tivoli Performance Viewer uses these interfaces to enable more comprehensive performance analysis.

JVMTI is a two-way function call interface between the JVM and an in-process profiler agent. The JVM notifies the profiler agent of various events, for example, garbage collection and thread starts. The profiler agent activates or deactivates specific event notifications that are based on the needs of the profiler.

JVMTI supports partial profiling by enabling you to choose which types of profiling information to collect and to select certain subsets of the time during which the JVM API is active. JVMTI moderately increases the performance impact. Therefore, it is good to use JVMTI monitoring to help diagnose application problems only.

To learn how to enable the Java virtual machine profiler data, read the information center article:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tprf_jvmpidata.html

OutOfMemory indicators in the javacore file

- Look to see whether the dump was due to an OutOfMemoryError

```
1TISIGINFO      Dump Event "systhrow" (00040000) Detail
"java/lang/OutOfMemoryError" received
```

- Check MEMINFO subcomponent output

0SECTION	MEMINFO subcomponent dump routine
NULL	=====
1STHEAPTOTAL	Total memory: 268435456 (0x10000000)
1STHEAPINUSE	Total memory in use: 268013560 (0x0FF98FF8)
1STHEAPFREE	Total memory free: 421896 (0x00067008)

- Check GC history

```
j9mm.133 - Allocation failure start: newspace=18080/56623104
oldspace=435936/201326592 loa=0/0 requestedbytes=1024008
```

- Free memory indicator

- If the free space is tight, increasing the size of the heap might solve the problem
- If there is much free space available, the problem might be due to a large object allocation or problem with native memory allocation

© Copyright IBM Corporation 2013

Figure 8-18. OutOfMemory indicators in the javacore file

WA5913.0

Notes:

Total memory is 256 MB (268435456).

Total memory free is 0.4 MB, nearly 0%.

Requested bytes are 0.98 MB.

An earlier slide explained that the javacore is the first place to check when investigating OutOfMemoryError artifacts. The javacore cannot confirm that the cause was in fact an OutOfMemoryError; it also includes information on heap use, recent GC activity, and the call stack of the running thread when the allocation was requested. This check can often rule out several of the potential causes for an OutOfMemoryError.



How to obtain a verboseGC log

- The JVM runtime provides the Verbose GC option
- Enables a garbage collection log
 - Interval between collections
 - Duration of collection
 - Whether compaction is required
 - Memory size, memory that is freed, memory available
- Enable the verbose GC for the server by using the administration console
 - **Servers > Server Types > WebSphere application servers > server_name**
 - Under Server Infrastructure, click **Java and Process Management > Process Definition > Java Virtual Machine**
 - Select **Verbose Garbage Collection** check box
 - Restart the server or configure on runtime tab
- Usually writes to `native_stderr`
 - Varies depending on the operating system and WebSphere version
 - Some performance impact because of disk I/O, but usually minimal unless thrashing

© Copyright IBM Corporation 2013

Figure 8-19. How to obtain a verboseGC log

WA5913.0

Notes:

The recent GC information in the javacore might not be enough to determine the cause of the OOM; therefore, it is a good practice for administrators to enable verbose GC for the Application Servers. In fact, several support teams encourage customers to always enable verbose GC because the added value of the information on the GC logs far exceeds the administrative cost of having to monitor the disk usage. VerboseGC can be easily enabled through the administrative console and requires a server restart to take effect. By default, the verboseGC output writes to the `native_stderr.log` file. There is some administrative cost due to writing the information to disk; however, this usage is minimal.

Default behavior for OutOfMemory exceptions

- Beginning with Version 8.0.0.2, WebSphere Application Server comes with IBM Java 6 R26 on supported operating systems
 - In this version, the default behavior for OutOfMemory (OOM) exceptions is changed
- By default, the first OOM for the lifetime of a Java process produces the following dumps:
 - PHD-formatted heap dump
 - Java dump file
 - Snap dump file
 - Operating system dump: **core** file on Linux, AIX, and IBM i, **user-mode minidump** with full memory on Windows operating systems, and **SYSTDUMP** on z/OS
- The second, third, and fourth OOM exceptions produce only a PHD-formatted heap dump and a Java dump file
- Therefore, the change in default behavior is an extra system dump on the first OOM exception

© Copyright IBM Corporation 2013

Figure 8-20. Default behavior for OutOfMemory exceptions

WA5913.0

Notes:

See the information center topic, “Default behavior for OutOfMemory exceptions”:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.base.doc%2Finfo%2Faes%2Fae%2Fctrb_java626.html

System dump files

- A system dump is a superset of a PHD heap dump
- A system dump also includes memory contents (strings, primitives, variable names, and others), thread and frame local information, some native memory information, and more
- This added information can solve a larger class of problems, provide more general insight into a running JVM, and ultimately reduce the time that it takes to solve a problem
- In earlier versions of IBM Java, a system dump had to be post-processed using the `jextract` tool
- With recent versions of IBM Java, a DTFJ-capable tool, such as the Memory Analyzer Tool, can directly load a system dump without any post-processing for OOMs
- For crashes, however, `jextract` should still be used

© Copyright IBM Corporation 2013

Figure 8-21. System dump files

WA5913.0

Notes:

For more information, go to:

<http://www.ibm.com/developerworksopensource/library/j-memoryanalyzer/index.html>

Obtaining system core dumps

- In addition to being automatically triggered the first time an OutOfMemory error occurs, system core dumps can be triggered manually in several ways:
 - Using wsadmin
 - Using the system dump button in the administrative console
 - Using OS commands
- Run the following wsadmin commands:

```
jvm=AdminControl.completeObjectName(  
    'type=JVM,<server_name>,*)  
AdminControl.invoke(jvm, 'generateSystemDump')
```

- This function is not supported when using a non-IBM Java virtual machine (JVM)
 - HP-UX and Solaris operating systems

© Copyright IBM Corporation 2013

Figure 8-22. Obtaining system core dumps

WA5913.0

Notes:

The sample script can trigger a system core dump with wsadmin. This method might not work if the target JVM exhausts the Java heap and is unable to process the message that instructs it to create the system core dump.

Java heap dumps

- A heap dump contains all the reachable objects that are on the Java heap
 - Those objects that the Java application uses
- Heap dumps are created by default for OutOfMemory exceptions
 - By default, heap dumps are in binary (PHD) format
 - Can also be generated in test format
- Used for analyzing the actual contents of the Java heap
 - Shows the objects that are using the heap memory
 - Shows the objects that are using large amounts of memory on the Java heap, and what is preventing the Garbage Collector from collecting them
 - Useful for debugging memory leaks in Java applications

© Copyright IBM Corporation 2013

Figure 8-23. Java heap dumps

WA5913.0

Notes:

Heap dump is an IBM JVM facility that generates a dump of all the reachable objects that are on the Java heap; that is, the objects that a Java application uses. This dump is called a heap dump. It shows the objects that are using large amounts of memory on the Java heap, and what is preventing the garbage collector from collecting these objects.

The heap dump contains two lines per object. The first line displays the address of the object, various flags, its size, and type information. The second line contains a list of the memory addresses of other objects that the object references.

Heap dumps versus system core dumps

- The Memory Analyzer tool accepts a full system core file or a heap dump file
 - Which is better?
- A system core dump allows you to do everything a heap dump can, however
 - It is considerably larger
 - It usually takes a little longer to create
- The size and time to create the system core must be balanced against its advantages, such as:
 - Accurate GC root information
 - Native heap memory
 - Ability to see the values of primitive fields and strings (might be a privacy issue)
 - More views such as threads and their related stacks and frame local references
- Unless the write time or size make system cores unmanageable, consider the use of them instead of heap dumps

© Copyright IBM Corporation 2013

Figure 8-24. Heap dumps versus system core dumps

WA5913.0

Notes:

You should be aware that since the values of some variables and strings are available in a system core dump, you might display confidential data such as credit card numbers and identity information. If you must protect the privacy of such data, system core files must be kept confidential.

Obtaining Java heap dumps (1 of 3)

- Generated from a running JVM in these ways:
 - Explicit, manual generation, such as using `wsadmin` commands
 - JVM-triggered generation, by 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 V1.4.2 and earlier: Use environment variables

© Copyright IBM Corporation 2013

Figure 8-25. Obtaining Java heap dumps (1 of 3)

WA5913.0

Notes:

The other useful artifact for determining the cause of an OOM is heap dumps. Heap dumps provide insight into the contents of the Java heap and are the best way to determine what might be causing a memory leak. Several methods can generate heap dumps, including `wsadmin`, JVM triggers, or automatically when an OOM or other conditions occur.

Obtaining Java heap dumps (2 of 3)

- Oracle JVM version 1.4.2_08 and earlier use IBM Heapagent
 - Not supported for production environments without assistance from IBM Support
- Oracle JVM version 1.4.2_09, 1.4.2_10, and 1.4.2_11 use JMapp
 - Command line tool that comes with the Oracle 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 that is used by all instances
 - Use the output operator to send output to a file
- Oracle JVM version 1.4.2_12 and higher allows Java heap dump
 - JVM command line option `-XX:+HeapDumpOnCtrlBreak`
 - SIGQUIT signal (`kill -3`) triggers the JVM to generate a heapdump
 - Heapdump can be parsed by using IBM Memory Analyzer

© Copyright IBM Corporation 2013

Figure 8-26. Obtaining Java heap dumps (2 of 3)

WA5913.0

Notes:

The method for creating a heap dump on a Solaris system varies depending on the version of the JVM and the specific fix level. It is generally best to contact IBM support to determine how to configure heap dump creation for your specific configuration.

See the Oracle J2SE 6.0 “Troubleshooting and Diagnostic Guide” for more information.

Obtaining Java heap dumps (3 of 3)

- Java memory dumps on HP-UX
 - HPROF profiler agent: Not useful with larger heap sizes, which are typically used with WebSphere
 - Jmap
- Work with IBM Support to find the best solution for your environment

© Copyright IBM Corporation 2013

Figure 8-27. Obtaining Java heap dumps (3 of 3)

WA5913.0

Notes:

Heap dump creation is even more involved for HP-UX systems and might not be possible with larger heap sizes. The best solution is to contact support and work with them to devise a solution for your environment.

For more information, see: <http://docs.hp.com/en/5992-4687/ch01s10.html>

Use wsadmin to trigger a heap dump

- From a command prompt, start the wsadmin shell by typing:

```
wsadmin -lang jython -user <user_name> -password <password>
```

- Run the following wsadmin commands:

```
jvm=AdminControl.completeObjectName(  
    'type=JVM,<server_name>,*')  
AdminControl.invoke(jvm,'generateHeapDump')
```

- This function is not supported when using a non-IBM Java virtual machine (JVM)
 - HP-UX and Solaris operating systems

© Copyright IBM Corporation 2013

Figure 8-28. Use wsadmin to trigger a heap dump

WA5913.0

Notes:

The sample script can trigger a heap dump with wsadmin. This method might not work if the target JVM exhausts the Java heap and is unable to process the message that instructs it to create the heap dump.

Java heap dumps by using dump agents

- To obtain a Java heap dump on a user signal, or if the application catches the resulting signal, add `-Xdump` command-line options to the Generic JVM arguments setting for a server
- For example, to create heap dumps, even when the application catches an `OutOfMemoryError`, use the argument

```
-Xdump:heap:events=throw,filter=java/lang/OutOfMemoryError
```



© Copyright IBM Corporation 2013

Figure 8-29. Java heap dumps by using dump agents

WA5913.0

Notes:

To access the Generic JVM arguments configuration field from the administrative console, select **Servers > Server Types > WebSphere Application Servers > server_name > Java and Process Management > Process Definition > Java Virtual Machine**.

A more reliable method of creating a heap dump on `OutOfMemoryError` is to use the `-Xdump` JVM argument for IBM JDK V5 and later. The option instructs the JVM to write a heap dump whenever the `OutOfMemoryError` occurs and can create the dump even if the Java heap is exhausted. There are several options for configuring the `-Xdump` option; however, any changes to the JVM argument take effect only after a restart.

For detailed description of `-Xdump` usage, see:

<http://www.ibm.com/support/docview.wss?uid=swg21242497>

Generic arguments that use Xdump:heap

- `-Xdump:heap:events=...` enables heap dump creation for specified events
- Events can be:
 - `user` (the JVM receives the SIGQUIT or SIGBREAK signal from the OS)
 - `vmstart, vmstop` (the virtual machine is started or stopped)
 - `throw` (an exception is thrown; use filter on exception class)
 - `systhrow` (the JVM is about to throw a Java exception)
 - Several other 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
- `-Xdump:heap:file=C:\dumps\heapdump.%Y%m%d.%H%M%S.%pid.%seq.phd` specifies the location and format of the heap dump file name

© Copyright IBM Corporation 2013

Figure 8-30. Generic arguments that use Xdump:heap

WA5913.0

Notes:

- `user`: The JVM receives the SIGQUIT (Linux, AIX, z/OS, and i5/OS) or SIGBREAK (Windows) signal from the operating system.
- `throw`: An exception is thrown.
- `systhrow`: The JVM is about to throw a Java exception. This event is different from the “throw” event because it is triggered only for error conditions that are detected internally in the JVM.
- `filter`: Filters on exception class name, for example: `filter=java/lang/OutOfMem*`
- Run the command `<was_root>/java/bin/java -version -Xdump:what` to see your JVM default agents.

Generic arguments that use Xdump:system

- `-Xdump:system:events=...` enables system dump creation for specified events
- Events can be:
 - gpf
 - abort
 - systhrow (the JVM is about to throw a Java exception)
 - Several other events
- `-Xdump:system:none` disable system core dump creation
- `-Xdump:system` enables system core dump creation
- `-Xdump:system:file=/usr/dumps/core.%Y%m%d.%H%M%S.%pid.%seq.dmp` specifies the location and format of the system core dump file name

© Copyright IBM Corporation 2013

Figure 8-31. Generic arguments that use Xdump:system

WA5913.0

Notes:

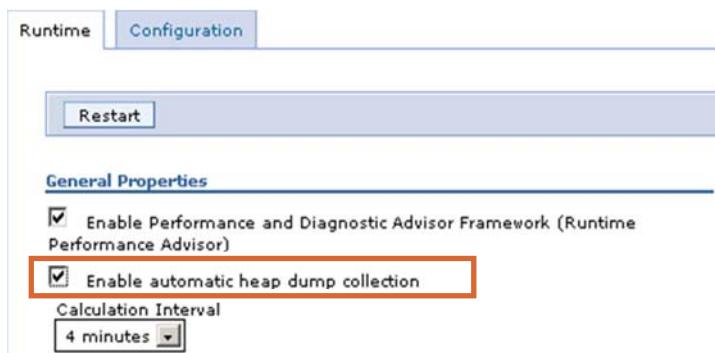
A gpf event is triggered when a General Protection Fault (GPF) occurs.

An abort event is triggered when the JVM receives the SIGABRT signal from the operating system.

A trace assert event is triggered when an internal error occurs in the JVM.

Automatic heap dump generation with IBM JDK V6

- Works only with IBM JVMs
- Works with the lightweight memory leak detection mechanism
- Heap dumps are generated when a memory leak is detected or when `OutOfMemoryError` is detected
 - False positives might occur with gencon and balanced GC policies
- Perform the following steps in the administrative console:
 1. Click **server_name > Performance and Diagnostic Advisor Configuration**
 2. Select the **Enable automatic heap dump collection** check box on the Runtime tab



© Copyright IBM Corporation 2013

Figure 8-32. Automatic heap dump generation with IBM JDK V6

WA5913.0

Notes:

Another option for generating heap dumps is to use the automatic dump generation that is part of IBM JDK V5 and newer. This method works with a memory leak detection mechanism that potentially generates the dump before the OOM but after the leak is detected. This option can be enabled through the administrative console.

Enable automatic heap dump collection specifies whether the Performance and Diagnostic Advisor automatically generates heap dumps for post analysis when suspicious memory activity is detected.

Suppose that you have a memory leak and want to confirm the leak, or you want to automatically generate heap dumps on Java virtual machines (JVM) in WebSphere Application Server. In this case, consider changing your minimum and maximum heap sizes to be equal. This change provides the memory leak detection more time for reliable diagnosis.

8.2. Interpret verbose GC output

Interpret verbose GC output



© Copyright IBM Corporation 2013

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

8.0

Figure 8-33. Interpret verbose GC output

WA5913.0

Notes:

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 failure an isolated allocation failure caused by a request for a large object?

© Copyright IBM Corporation 2013

Figure 8-34. OutOfMemory: Interpret verboseGC output

WA5913.0

Notes:

VerboseGC logs contain a great deal of information about how the JVM is using the Java heap space. The first step when analyzing verboseGC logs for an OOM is to find the allocation request that caused the OOM and check the size of the allocation. If the allocation is large (again assume anything over 1 MB is approaching large), then a large object collection likely caused the OOM. The next step is to determine the total size of the heap and what percentage is used. This process might be tricky if the JVM is using generational collection because the heap is divided into multiple sections and therefore the free spaces are not all part of the same resource pool.

Another option is to look at past allocation failures and see whether the free space after collection is progressively decreasing in size or if there are other large allocations in the past. This analysis can be done by reviewing the log file manually; however, graphing tools are likely to prove easier to use.

JVM messages that show dump events

- The IBM JDK will log messages to native_stderr.log

```
JVMDUMP039I Processing dump event "systhrow", detail
"java/lang/OutOfMemoryError" at 2012/03/15 15:51:49 - please wait.

JVMDUMP032I JVM requested System dump using
'C:\Sysdumps\core.20120315.155149.412.0002.dmp' in response to an event
JVMDUMP010I System dump written to
C:\Sysdumps\core.20120315.155149.412.0002.dmp
JVMDUMP032I JVM requested Heap dump using
'C:\Dumps\heapdump.20120315.155149.412.0003.phd' in response to an event
JVMDUMP010I Heap dump written to
C:\Dumps\heapdump.20120315.155149.412.0003.phd
JVMDUMP032I JVM requested Java dump using
'C:\Dumps\javacore.20120315.155149.412.0004.txt' in response to an event
JVMDUMP010I Java dump written to
C:\Dumps\javacore.20120315.155149.412.0004.txt
JVMDUMP032I JVM requested Snap dump using 'C:\Program
Files\IBM\WebSphere\AppServer\profiles\profile1\Snap.20120315.155149.412
.0005.trc' in response to an event

JVMDUMP013I Processed dump event "systhrow", detail
"java/lang/OutOfMemoryError".
```

© Copyright IBM Corporation 2013

Figure 8-35. JVM messages that show dump events

WA5913.0

Notes:

The IBM JDK logs messages to native_stderr.log to indicate when a heap dump is created because of a dump event. In this case, the dump event is the OutOfMemoryError.

Starting in WebSphere Application Server 8.0.0.2, the IBM Java SDK changed the default OutOfMemoryError agents to produce a system dump on the first OOM in addition to the previous defaults. This change is by design.

Run the command <was_root>/java/bin/java -version -Xdump:what to see your JVM default agents and search for those lines that have filter=java/lang/OutOfMemoryError.

Snap dumps contain the tracepoint data that is held in the trace buffers. Snap traces require the use of the trace formatter for further analysis.

Verbose GC output for optthruput policy

```

<af type="tenured" id="1" timestamp="Sun Mar 12 19:12:55 2006" intervalms="0.000">
  <minimum requested_bytes="16" />
  <time exclusiveaccessms="0.025" />
  <tenured freebytes="23592960" totalbytes="471859200" percent="5" >
    <soa freebytes="0" totalbytes="448266240" percent="0" />
    <loa freebytes="23592960" totalbytes="23592960" percent="100" />
  </tenured>
  <gc type="global" id="3" totalid="3" intervalms="11620.259">
    <refs_cleared soft="0" weak="72" phantom="0" />
    <finalization objectsqueued="9" />
    <timesms mark="74.734" sweep="7.611" compact="0.000" total="82.420" />
    <tenured freebytes="409273392" totalbytes="471859200" percent="86" >
      <soa freebytes="385680432" totalbytes="448266240" percent="86" />
      <loa freebytes="23592960" totalbytes="23592960" percent="100" />
    </tenured>
  </gc>
  <tenured freebytes="409272720" totalbytes="471859200" percent="86" >
    <soa freebytes="385679760" totalbytes="448266240" percent="86" />
    <loa freebytes="23592960" totalbytes="23592960" percent="100" />
  </tenured>
  <time totalms="83.227" />
</af>

```

© Copyright IBM Corporation 2013

Figure 8-36. Verbose GC output for optthruput policy

WA5913.0

Notes:

This example shows verbose GC output that uses the optthruput policy, which was the default policy for a WebSphere 7.0 Application Server.

Java 6.26 GC output: optthrput policy (1 of 3)

```

<exclusive-start id="44" timestamp="2012-02-24T10:09:28.704" intervalms="2859.607">
  <response-info timems="0.017" idlems="0.017" threads="0" lastid="00149A00"
    lastname="main" />
</exclusive-start>
  Request that caused
  allocation failure
<af-start id="45" totalBytesRequested="10016" tsT:09.704" intervalms="2859.783" />
<cycle-start id="46" type="global" contextid="0" tsT:09.704" intervalms="2859.852" />
<gc-start id="47" type="global" contextid="46" timestamp="2012-02-24T10:09.720">
  <mem-info id="48" free="2669000" total="52428800" percent="5">
    <mem type="tenure" free="2669000" total="52428800" percent="5">
      <mem type="soa" free="47560" total="49807360" percent="0" />
      <mem type="loa" free="2621440" total="2621440" percent="100" /> </mem>
    </mem-info>
  </gc-start>
  Small object area (soa) has
  0% free
<allocation-stats totalBytes="27037240" >
  <allocated-bytes non-tlh="9534880" tlh="17502360" />
  <largest-consumer threadName="main" threadId="00149A00" bytes="26956888" />
</allocation-stats>

```

© Copyright IBM Corporation 2013

Figure 8-37. Java 6.26 GC output: optthrput policy (1 of 3)

WA5913.0

Notes:

In Java 6.26 the format for verbose GC data is changed. The next several slides show what the new format is for optthrput and gencon modes.

See the topic “Verbose garbage collection logging” for details:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/topic/com.ibm.java.doc.60_26/vm626/J9/GC/gcpd_verbosegc.html

Stop-the-world operations

When an application is stopped so that the garbage collector has exclusive access to the JVM, verbose logging records the event.

Stop-the-world operations are shown within the `<exclusive-start>` and `<exclusive-end>` tags. The `<exclusive-start>` tag includes the response-info section, which provides details about the process of acquiring exclusive access to the JVM.

Garbage collection cycle

Verbose garbage collection output shows each garbage collection cycle that is enclosed within `<cycle-start>` and `<cycle-end>` tags.

The `<cycle-end>` tag contains a `context-id` attribute that indicates the ID of the corresponding `<cycle-start>` tag.

Garbage collection increment

A complete garbage collection increment is shown within `<gc-start>` and `<gc-end>` tags in the verbose output.

The `<gc-start>` and `<gc-end>` tags represent the start and end of a garbage collection increment. Both of these tags include a mem-info section. This section includes information about the current state of the Java heap, and any memory that a specific garbage collection policy uses.

The `<gc-start>` and `<gc-end>` tags contain a `context-id` attribute that indicates the ID of the corresponding garbage collection cycle.

Allocation failure

The `<af-start>` and `<af-end>` tags show garbage collection cycles that an allocation failure causes in the verbose output.

The `<af-start>` and `<af-end>` tags enclose the `<cycle-start>` and `<cycle-end>` tags. The `<af-start>` tag contains a `totalBytesRequested` attribute. This attribute specifies the number of bytes that the allocations that caused this allocation failure required. The `intervalms` attribute on the `<af-start>` tag is the time, in milliseconds, since the previous `<af-start>` tag. When the garbage collection cycle caused by the allocation failure is complete, an `allocation-satisfied` tag is generated. This tag indicates that the allocation that caused the failure is now complete.

Java 6.26 GC output: optthrput policy (2 of 3)

```

<gc-op id="49" type="mark" timems="63.254" contextid="46" timestamp="2012-02-
24T10:09:28.767">
    GC operation = Mark phase
    <trace-info objectcount="325795" scancount="260990" scanbytes="8497304" />
    <finalization candidates="659" enqueueued="78" />
    <references type="soft" candidates="3271" cleared="0" enqueueued="0"
dynamicThreshold="18" maxThreshold="32" />
    <references type="weak" candidates="12682" cleared="0" enqueueued="0" />
    <references type="phantom" candidates="1" cleared="0" enqueueued="0" />
    <stringconstants candidates="21495" cleared="100" />
</gc-op>

<gc-op id="50" type="classunload" timems="0.011" contextid="46" timestamp="2012-02-
24T10:09:28.767">
    GC operation = Classunload phase
    <classunload-info classloadercandidates="85" classloadersunloaded="0"
classesunloaded="0" quiescems="0.000" setupms="0.007" scanms="0.002"
postms="0.002" />
</gc-op>
```

© Copyright IBM Corporation 2013

Figure 8-38. Java 6.26 GC output: optthrput policy (2 of 3)

WA5913.0

Notes:

Garbage collection operation

Every garbage collection increment contains at least one garbage collection operation, which is shown in the verbose output with a `<gc-op>` tag.

The `timems` attribute of the `<gc-op>` tag indicates the time that is taken to complete the garbage collection operation, in milliseconds. The `contextid` attribute indicates the ID of the corresponding garbage collection cycle. In the following example, `contextid="46"` indicates that this garbage collection operation is part of the garbage collection cycle that has the tag `<cycle-start id="46">`.

The `<gc-op>` output contains subsections that describe operations that are specific to different garbage collection policies. These subsections might change from release to release, when improvements are made to the technology or when new data becomes available.

Java 6.26 GC output: optthruput policy (3 of 3)

```

<gc-op id="51" type="sweep" timems="2.933" contextid="46" timestamp="2012-02-24T10:09:28.767" />
    GC operation = sweep phase

<gc-end id="52" type="global" contextid="46" durationms="68.258" timestamp="2012-02-24T10:09:28.767">

<mem-info id="53" free="26769528" total="52428800" percent="51">
    <mem type="tenure" free="26769528" total="52428800" percent="51">
        <mem type="soa" free="24148088" total="49807360" percent="48" />
            <mem type="loa" free="2621440" total="2621440" percent="100" />
                48% of soa recovered
    </mem>
    <pending-finalizers system="77" default="1" reference="0" classloader="0" />
</mem-info>
</gc-end>
    Memory request was satisfied

<cycle-end id="54" type="global" contextid="46" timestamp="2012-02-24T10:09:28.767" />
<allocation-satisfied id="55" threadId="00149A00" bytesRequested="10016" />
<af-end id="56" timestamp="2012-02-24T10:09:28.767" />
<exclusive-end id="57" timestamp="2012-02-24T10:09:767" durationms="74.224" />

```

© Copyright IBM Corporation 2013

Figure 8-39. Java 6.26 GC output: optthruput policy (3 of 3)

WA5913.0

Notes:

The `<cycle-end>` tag contains a `context-id` attribute that indicates the ID of the corresponding `<cycle-start>` tag.

In the example that is shown, the `<cycle-end>` tag has a `context-id` of 46, which reflects the ID value that is shown for `<cycle-start>`.

When the garbage collection cycle caused by the allocation failure is complete, an `allocation-satisfied` tag is generated. This tag indicates that the allocation that caused the failure is now complete.

Verbose GC output for gencon policy

```

<af type="nursery" id="35" timestamp="Thu Aug 11 21:47:11 2005"
  intervalms="10730.361">
  <minimum requested_bytes="144" />
  <time exclusiveaccessms="1.193" />
  <nursery freebytes="0" totalbytes="1226833920" percent="0" />
  <tenured freebytes="68687704" totalbytes="209715200" percent="32" />
    <soa freebytes="58201944" totalbytes="199229440" percent="29" />
    <loa freebytes="10485760" totalbytes="10485760" percent="100" />
  </tenured>
  <gc type="scavenger" id="35" totalid="35" intervalms="10731.054">
    <flipped objectcount="1059594" bytes="56898904" />
    <tenured objectcount="12580" bytes="677620" />
    <refs_cleared soft="0" weak="691" phantom="39" />
    <finalization objectsqueued="1216" />
    <scavenger tiltratio="90" />
    <nursery freebytes="1167543760" totalbytes="1226833920" percent="95" />
    tenureage="14" />
    <tenured freebytes="67508056" totalbytes="209715200" percent="32" />
      <soa freebytes="57022296" totalbytes="199229440" percent="28" />
      <loa freebytes="10485760" totalbytes="10485760" percent="100" />
    </tenured>
    <time totalms="368.309" />
  </gc>
  <nursery freebytes="1167541712" totalbytes="1226833920" percent="95" />
  <tenured freebytes="67508056" totalbytes="209715200" percent="32" />
    <soa freebytes="57022296" totalbytes="199229440" percent="28" />
    <loa freebytes="10485760" totalbytes="10485760" percent="100" />
  </tenured>
  <time totalms="377.634" />
</af>

```

Allocation request details: id, when and time between AF

Heap occupancy details before GC

Details about the scavenge

Heap occupancy details after GC

© Copyright IBM Corporation 2013

Figure 8-40. Verbose GC output for gencon policy

WA5913.0

Notes:

This sample verboseGC log shows the gencon collection policy, which is the new default policy in WebSphere Application Server V8. Notice how there is now a nursery section and more information about how objects were transferred from nursery to tenured; however, the same allocation size and timing information are still present.

Java 6.26 GC output: gencon policy (1 of 3)

```

<exclusive-start id="26" timestamp="2012-02-23T20:53:38.440"
    intervalms="251.038">
    <response-info timems="0.010" idlems="0.010" threads="0"
        lastid="0014A400" lastname="main" />
</exclusive-start>                                Request that caused allocation failure

<af-start id="27" totalBytesRequested="25424" timestamp="2012-02-
    23T20:53:38.440" intervalms="250.537" />
<cycle-start id="28" type="scavenge" contextid="0" timestamp="2012-02-
    23T20:53:38.440" intervalms="250.497" />
<gc-start id="29" type="scavenge" contextid="28" timestamp="2012-02-
    23T20:53:38.440">                            GC operation = scavenge

    <mem-info id="30" free="38928232" total="45875200" percent="84">
        <mem type="nursery" free="0" total="6553600" percent="0" />
        <mem type="tenure" free="38928232" total="39321600" percent="98">
            <mem type="soa" free="36962152" total="37355520" percent="98" />
            <mem type="loa" free="1966080" total="1966080" percent="100" />
        </mem>
        <remembered-set count="5703" />
    </mem-info>
</gc-start>                                Nursery has 0 memory free
                                                before the scavenge

```

© Copyright IBM Corporation 2013

Figure 8-41. Java 6.26 GC output: gencon policy (1 of 3)

WA5913.0

Notes:

This allocation request for 25424 bytes failed because the nursery has 0 free bytes.

Java 6.26 GC output: gencon policy (2 of 3)

```

<allocation-stats totalBytes="4002592" >
  <allocated-bytes non-tlh="21376" tlh="3981216" />
  <largest-consumer threadName="main" threadId="0014A400" bytes="4002592" />
</allocation-stats>
<gc-op id="31" type="scavenge" timems="8.391" contextid="28"
  timestamp="2012-02-23T20:53:38.440">
  GC op scavenge begins
  <scavenger-info tenureage="9" tiltratio="50" />
  <memory-copied type="nursery" objects="41652" bytes="2148492"
    bytesdiscarded="392" />
  <finalization candidates="118" enqueueued="0" />
  <references type="soft" candidates="36" cleared="0" enqueueued="0"
    dynamicThreshold="32" maxThreshold="32" />
  <references type="weak" candidates="21" cleared="0" enqueueued="0" />
  <references type="phantom" candidates="1" cleared="0" enqueueued="0" />
</gc-op>
<gc-end id="32" type="scavenge" contextid="28" durationms="9.100"
  timestamp="2012-02-23T20:53:38.440">

```

© Copyright IBM Corporation 2013

Figure 8-42. Java 6.26 GC output: gencon policy (2 of 3)

WA5913.0

Notes:

Details of the scavenge are reported. The tenure age is 9, meaning that an object must survive nine scavenges before it is promoted to the tenured space. The tilt ratio is 50, meaning that the allocate and survivor spaces are equal in size. In this example, 41652 objects with a total size of 2,148,492 bytes were copied to the survivor space.

Java 6.26 GC output: gencon policy (3 of 3)

```

<mem-info id="33" free="43171168" total="45875200" percent="94">
    <mem type="nursery" free="4242936" total="6553600" percent="64" />
    <mem type="tenure" free="38928232" total="39321600" percent="98">
        <mem type="soa" free="36962152" total="37355520" percent="98" />
        <mem type="loa" free="1966080" total="1966080" percent="100" />
    </mem>
    <remembered-set count="2743" />
</mem-info>

</gc-end>

<cycle-end id="34" type="scavenge" contextid="28" timestamp="2012-02-23T20:53:38.456" />

<allocation-satisfied id="35" threadId="0014A400" bytesRequested="25424" />
<af-end id="36" timestamp="2012-02-23T20:53:38.456" />
<exclusive-end id="37" timestamp="2012-02-23T20:53:38.456" durationms="13.062" />

```

Scavenge recovers 64% of the nursery

Memory request was satisfied

© Copyright IBM Corporation 2013

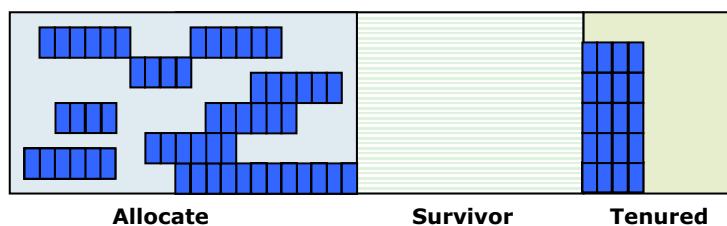
Figure 8-43. Java 6.26 GC output: gencon policy (3 of 3)

WA5913.0

Notes:

The scavenge recovered 64% of the nursery and allows the allocation request to succeed.

Verbose GC output for gencon failures



- Scavenging can fail due to a complete lack of space.
 - The scavenge is stopped and a global collect is attempted
 - Message in verbose GC log is: warning details="aborted collection"
- Nursery collections sometimes determine that they are insufficient.
 - Failure to complete a nursery collect
 - Insufficient resources
- Collect is promoted to a global GC.
 - Message in verbose GC log is: percolating_collect reason="insufficient remaining tenure space"

© Copyright IBM Corporation 2013

Figure 8-44. Verbose GC output for gencon failures

WA5913.0

Notes:

Generational garbage collection is a more advanced method of heap space management and therefore might seem more complex to understand. The important fact is that all allocations try to occur in the nursery region before anything occurs in the tenured region. If the allocation fails for the nursery (for whatever reason), then a global collection occurs and the object looks for space in either the nursery or tenured regions. If there is not adequate space in either location, an OOM occurs.

IBM Monitoring and Diagnostic Tools for Java – Garbage Collection and Memory Visualizer (GCMV)

GCMV: 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 use output that is related to application
 - Build plug-able analyzers for specific application needs

© Copyright IBM Corporation 2013

Figure 8-45. IBM Monitoring and Diagnostic Tools for Java – Garbage Collection and Memory Visualizer (GCMV)

WA5913.0

Notes:

Analyzing verboseGC logs is the best way to determine whether an OOM occurred due to a leak, especially if the leak is slow and requires an extensive amount of time to exhaust the heap space. GCMV is an excellent tool for analyzing verboseGC logs because it provides both an analysis summary and a robust graphing feature to visualize the past garbage collections.



Garbage Collection and Memory Visualizer overview

- The Garbage Collection and Memory Visualizer (GCMV) is a visualizer for verbose garbage collection output
 - The tool parses and plots verbose GC output and garbage collection traces (-Xtgc output)
- Started from the IBM Support Assistant
- The GCMV provides:
 - Raw view of data
 - Line plots to visualize various GC data characteristics
 - Tabulated reports with heap occupancy recommendations
 - View of multiple data sets on a single set of axes
 - Ability to save data as an image (jpeg) or comma-separated file (csv)

© Copyright IBM Corporation 2013

Figure 8-46. Garbage Collection and Memory Visualizer overview

WA5913.0

Notes:

GCMV is available through the IBM Support Assistant and can parse both verboseGC logs and garbage collection trace. The main value of GCMV is the ability to visualize the variety of GC data and visualize the health of the JVM over time. You can also export information from GCMV in .jpeg or .cvs format to share the findings with other parties.

GCMV usage scenarios

- Investigate performance problems
 - Long periods of pausing or not responding
- 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

© Copyright IBM Corporation 2013

Figure 8-47. GCMV usage scenarios

WA5913.0

Notes:

OOM is not the only reason to evaluate verboseGC logs. The information in verboseGC and help improve performance by finding a better heap size, more appropriate garbage collection policy, or by monitoring the growth in memory consumption.

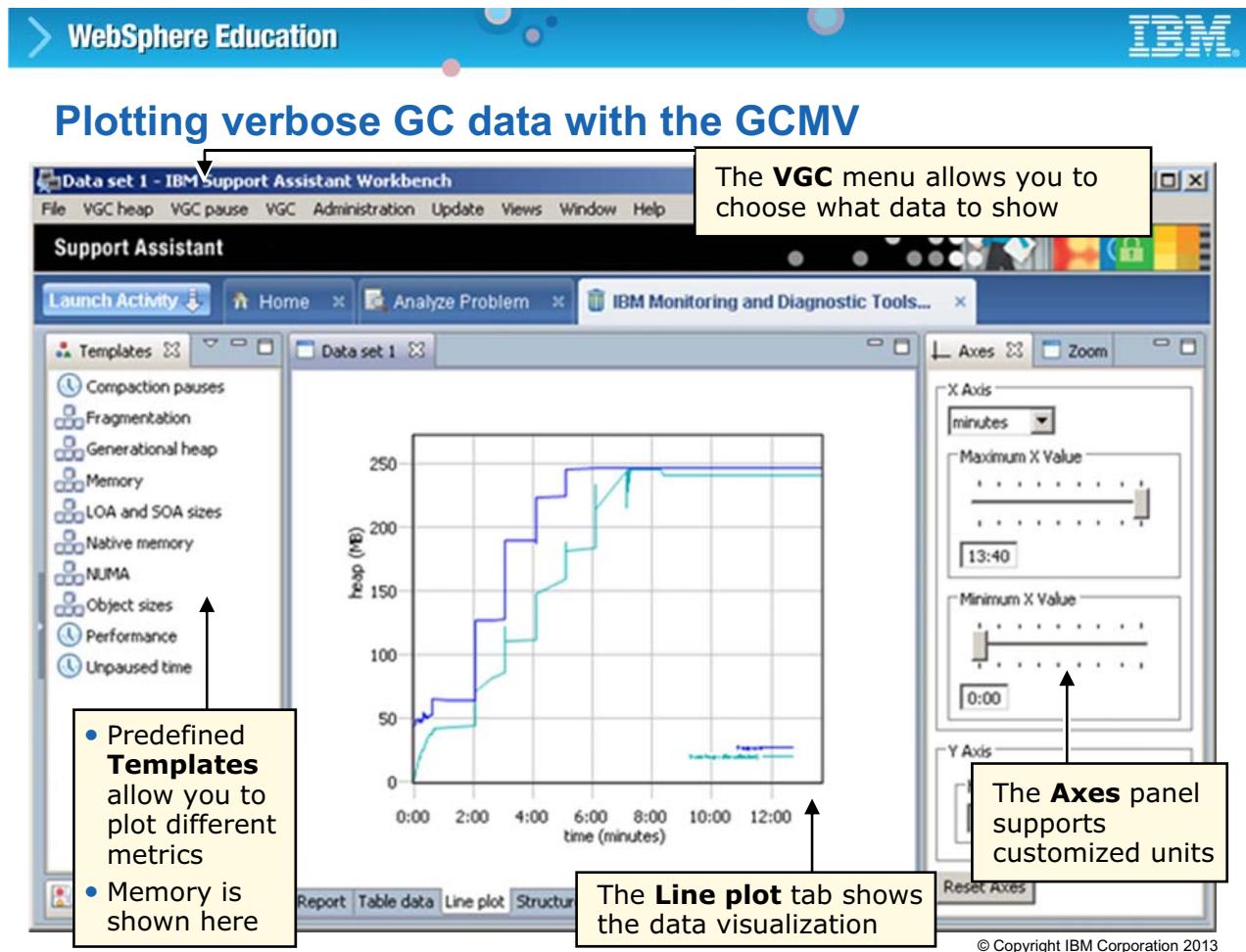


Figure 8-48. Plotting verbose GC data with the GCMV

WA5913.0

Notes:

Templates

The IBM Monitoring and Diagnostic Tools for Java - Garbage Collection and Memory Visualizer (GCMV) provides the following templates, which cover a range of common collections of data types to plot:

- **Memory:** Plots Heap size, Used heap (after collection), and JVM restarts. Useful for diagnosing memory leaks or poor heap sizing.
- **Performance:** Plots Mark, Sweep, Pause, Compact times, and JVM restarts. Useful for investigating performance issues.
- **Unpaused Time:** Plots Intervals between garbage collection triggers and Pause times. This template can help determine whether an application is spending long periods of time in garbage collection rather than running.
- **Object Sizes:** Plots Heap size, Used heap (after collection), and Requested object sizes that trigger allocation failures. This template can help determine whether an application is requesting large objects and triggering excessive garbage collection.

- **LOA and SOA Sizes:** Plots Heap size, Used SOA (small object area), Used LOA (large object area), Used tenured heap, and Used nursery heap (after collection). This template can help determine whether there is a problem with the `-Xlratio` setting.
- **Generational Heap:** Plots Heap size, Free heap, Tenured heap size, Free tenured heap, Nursery size, Free nursery heap, and Tenure age. Useful for diagnosing memory leaks or poor nursery or tenured heap sizings.
- **Fragmentation:** Plots Heap size, Free heap (before collection), and Free heap (after collection). A large amount of free heap before collection might indicate heap fragmentation. A small amount of free heap after collection might indicate that the heap size is too small or the application is holding onto too much data.
- **Compaction Pauses:** Plots Heap size, Used heap (after collection), Compact times, and Pause times. This template can help identify whether a restricted heap is causing excessive time that is spent in compaction.
- **Native Memory:** Plots native memory usage. Useful for diagnosing out-of-memory errors.
- **NUMA:** Non-uniform memory architecture information.

Types of graphs

The GCMV has built-in support for over 40 different types of graphs

- These graphs are configured in the VGC Data, VGC Pause Data, and VGC Heap Data menus
- 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 that are cleared
 - Soft references that are cleared
 - Free tenured heap (after collection)
 - Tenured heap size
 - Tenure age
 - Free LOA (after collection)
 - Free SOA (after collection)
 - Total LOA
 - Total SOA

© Copyright IBM Corporation 2013

Figure 8-49. Types of graphs

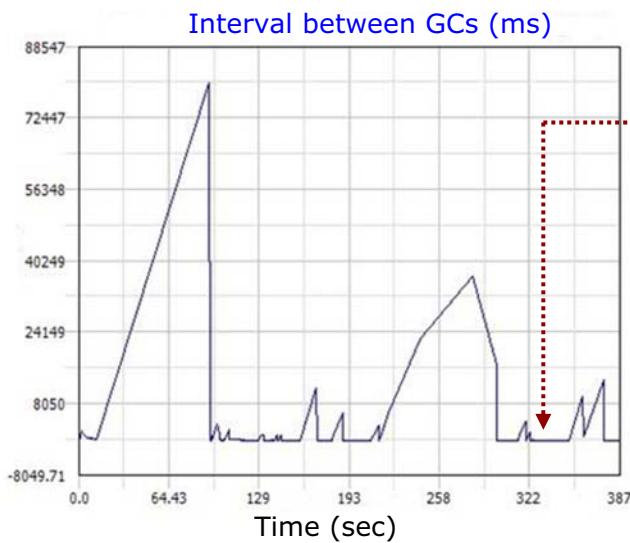
WA5913.0

Notes:

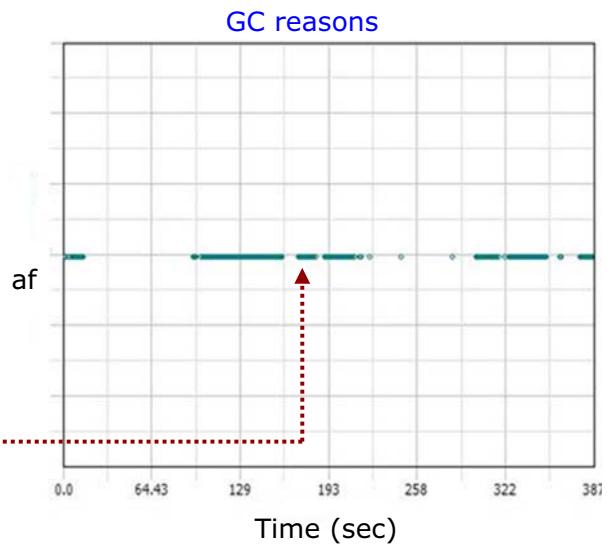
GCMV supports multiple types of graphs that correspond to the key metrics associated with Java heap management. Each graph type can be combined with any number of other types to compare trends across metrics.

Different graph types and a different menu are available for TGC (trace garbage collector `-Xtgc`) output.

Garbage collection trigger graphs



- This graph shows intervals between garbage collection cycles
- Notice that it shrinks to near 0 ms for extended periods (horizontal portions of the graph)



© Copyright IBM Corporation 2013

Figure 8-50. Garbage collection trigger graphs

WA5913.0

Notes:

This slide shows two different graphs that are produced from the same verbose GC data. The graph on the left is showing time intervals between garbage collection cycles; notice that it shrinks to near 0 ms for extended periods. The graph on the right is showing GC reasons.

Each dot in this graph represents a garbage collection cycle. All of these cycles ran for the same reason: allocation failure. Notice that the dot concentration lines up with the interval graph.

Heap usage and occupancy recommendation (1 of 2)

The summary report shows that the mean heap occupancy is 50% and that the application is using large objects

- ✖ Your application appears to be leaking memory. This is indicated by the used heap increasing at a greater rate than the application workload (measured by the amount of data freed).
- ✖ The Java Heap has been exhausted, leading to an out of memory error. You should consider increasing the Java Heap size using -Xmx if space allows. You can analyse the usage of the Java Heap for a memory leak by using the ISA Tool Add-on, IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer.
- ⚠ The application seems to be using some quite large objects. The largest request which triggered an allocation failure (and was recorded in the verbose gc log) was for 5120008 bytes.
- ⚠ 8% of allocation failures were caused by large allocations. Consider using balanced mode.
- ⚠ 430 global garbage collects took on average 2,570% longer than the average nursery collect. If you consider this abnormally high and unacceptable, consider using balanced mode.
- ⓘ The mean occupancy in the nursery is 50% which is close to optimal.

© Copyright IBM Corporation 2013

Figure 8-51. Heap usage and occupancy recommendation (1 of 2)

WA5913.0

Notes:

There are three levels of messages: Fatal (red X), Warning (yellow !), and Information (i).

For this particular example, this slide shows only a sample of the tuning recommendations that the tool provided. There were many more warning level messages.

In this example, the fatal messages are most relevant for troubleshooting an OutOfMemoryError. The tool detects that the application is leaking memory and exhausting the heap.

Heap usage and occupancy recommendation (2 of 2)

- Based on the usage summary that is shown, the tool suggests:
 - Increasing the nursery size
 - Increasing the tenure age

Summary

Concurrent collection count	7
Forced collection count	4
GC Mode	gencon
Global collections - Mean garbage collection pause (ms)	293
Global collections - Mean interval between collections (minutes)	1.63
Global collections - Number of collections	437
Global collections - Total amount tenured (MB)	101416
Largest memory request (bytes)	5120008
Number of collections triggered by allocation failure	1086
Nursery collections - Mean garbage collection pause (ms)	11.5
Nursery collections - Mean interval between collections (ms)	82267
Nursery collections - Number of collections	660
Nursery collections - Total amount flipped (MB)	1761
Nursery collections - Total amount tenured (MB)	662
Proportion of time spent in garbage collection pauses (%)	0.27
Proportion of time spent unpause (%)	99.73
Rate of garbage collection (MB/minutes)	10.1

© Copyright IBM Corporation 2013

Figure 8-52. Heap usage and occupancy recommendation (2 of 2)

WA5913.0

Notes:

Based on the usage summary, the tool offers the following tuning advice:

- The mean occupancy in the tenured area is 96%, which is high. You can improve application performance by increasing the nursery size, or increasing the tenure age.
- 36% of the nursery collects percolated to become global collects. This situation is affecting performance and increasing pause times. Consider the use of balanced mode.

Comparing verbose GC between different JVMs (1 of 2)

- Multiple verbose GC logs can be imported into GCMV and compared

Variant	OOM.log	PBW.log
Concurrent collection count	6	0
Forced collection count	0	0
GC Mode	gencon	gencon
Global collections - Mean garbage collection pause (ms)	313	166
Global collections - Mean interval between collections (minutes)	0.16	0.91
Global collections - Number of collections	47	1
Global collections - Total amount tenured (MB)	9515	46.4
Largest memory request (bytes)	5120008	392416
Number of collections triggered by allocation failure	75	48
Nursery collections - Mean garbage collection pause (ms)	16.1	11.9
Nursery collections - Mean interval between collections (ms)	13670717	9878838
Nursery collections - Number of collections	34	47
Nursery collections - Total amount flipped (MB)	338	96.6
Nursery collections - Total amount tenured (MB)	159	7.09
Proportion of time spent in garbage collection pauses (%)	3.31	0.28
Proportion of time spent unpause (%)	96.69	99.72
Rate of garbage collection (MB/minutes)	145	224

© Copyright IBM Corporation 2013

Figure 8-53. Comparing verbose GC between different JVMs (1 of 2)

WA5913.0

Notes:

This screen capture shows a comparative analysis of two different verbose GC data sets. You can do different kinds of comparisons. In this example, one GC data set (PBW.log) was from a server instance that was running normally, and the other GC data set (OOM.log) was from a server instance that experienced an out-of-memory condition.

Also, notice that both instances of the server were using the same GC mode gencon. You can also load test a server by using two different GC policies; for example, optimize for throughput and gencon. Then, you can have GCMV do a comparative analysis of the verbose GC data for each policy.

Comparing verbose GC between different JVMs (2 of 2)

Variant	Tuning recommendation
OOM.log	<ul style="list-style-type: none"> ✖ The Java Heap has been exhausted, leading to an out of memory error. You should consider increasing the Java Heap size using -Xmx if space allows. You can analyse the usage of the Java Heap for a memory leak by using the ISA Tool Add-on, IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer. ⚠ The application seems to be using some quite large objects. The largest request which triggered an allocation failure (and was recorded in the verbose gc log) was for 5120008 bytes.
PBW.log	<ul style="list-style-type: none"> ⚠ 46 global garbage collects took on average 1,969% longer than the average nursery collect. If you consider this abnormally high and unacceptable, consider using balanced mode. ℹ The mean occupancy in the nursery is 52% which is close to optimal. ⚠ 1 global garbage collects took on average 1,387% longer than the average nursery collect. If you consider this abnormally high and unacceptable, consider using balanced mode. ✓ The mean occupancy in the nursery is 10%. This is low, so the gencon policy is probably an optimal policy for this workload. ℹ The mean occupancy in the tenured area is 69% which is close to optimal.

© Copyright IBM Corporation 2013

Figure 8-54. Comparing verbose GC between different JVMs (2 of 2)

WA5913.0

Notes:

This screen capture shows the tuning recommendations for the different JVMs that were compared in the previous slide. The one with the out-of-memory condition has a mean occupancy of 52%; however the heap is exhausted. The recommendation is to increase the heap size, and to consider trying the “balanced” GC mode.



IBM Pattern Modeling and Analysis Tool (PMAT)

- Predates GCMV but can also be useful when analyzing verbose GC
 - Available from the IBM Support Assistant
 - Works with IBM JVM 1.3 and up
- Provides the following features:
 - Tabulated and graphical views
 - Summary usage analysis
 - Analysis includes tuning recommendations
- Suggestion: GCMV is your first choice; use PMAT only if you have some particular reason to prefer it

© Copyright IBM Corporation 2013

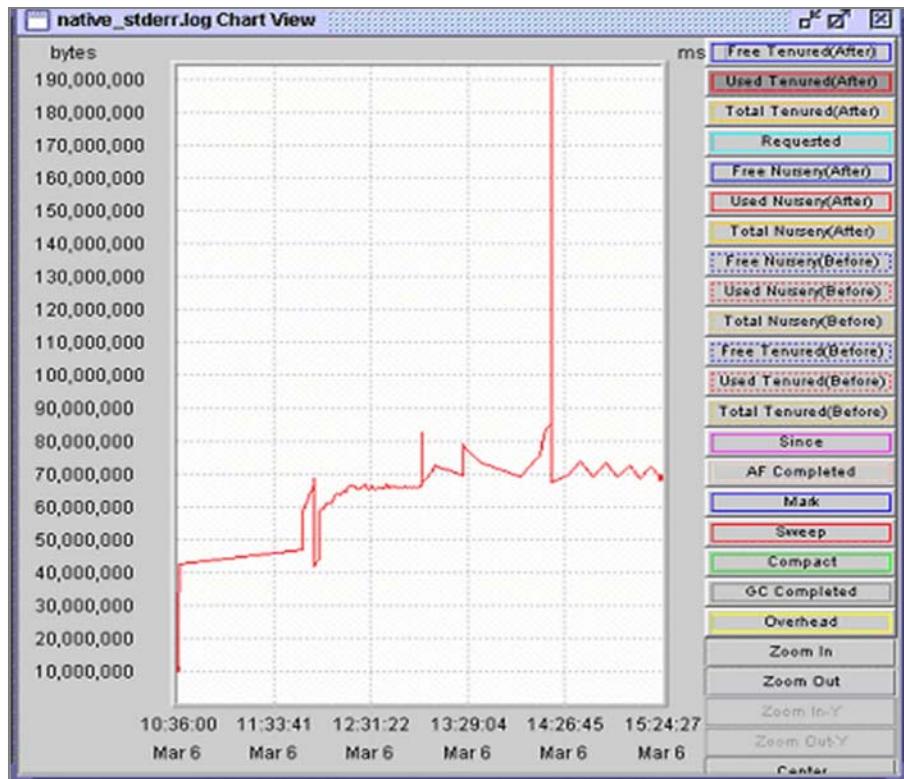
Figure 8-55. IBM Pattern Modeling and Analysis Tool (PMAT)

WA5913.0

Notes:

Another tool that predates GCMV but can also be useful when analyzing verboseGC is PMAT. PMAT provides similar capabilities to GCMV though it is not Eclipse-based.

PMAT: Chart view



© Copyright IBM Corporation 2013

Figure 8-56. PMAT: Chart view

WA5913.0

Notes:

The PMAT chart view allows you to create a graph of whatever GC data you like. In this example, the used heap space is graphed. Notice that other data types can be added to the chart as preferred. You can select the data that you want from the panel on the left. Notice that the spike in the used heap that is shown in this graph indicates a possible out-of-memory condition.

8.3. Analyze Java heap dumps and system core dumps

Analyze Java heap dumps and system core dumps



© Copyright IBM Corporation 2013

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

8.0

Figure 8-57. Analyze Java heap dumps and system core dumps

WA5913.0

Notes:

OutOfMemory: Interpret heap dumps

Heapdump:

- Use tools to parse the object trees
 - Almost impossible to do manually
- Look for root objects that hold significant memory resources
 - Includes memory that is allocated for the root object themselves and all the memory that is required for all the objects in the reference tree
- Check reference tree for sudden drop in memory
 - Expand the tree; 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
 - A long chain of similarly sized objects also indicates a potential leak

Jmap:

- Contact support to parse output
- Manually inspect the output list

© Copyright IBM Corporation 2013

Figure 8-58. OutOfMemory: Interpret heap dumps

WA5913.0

Notes:

The heap dump is the final artifact that is associated with an OOM and can be the most useful if you are trying to determine the cause of a memory leak. The heap dump contains the object hierarchy for every object in the Java heap. The normal approach to leak detection is to look for drops in the reach size of an object. The reach size (also referred to by other names) is the size of the current object except the cumulative size of all the objects beneath it in the hierarchy. A drastic drop in the reach size would indicate that the parent object has several multiple child objects that are not large on their own but add up to a substantial part of the heap. This method of analysis does not detect all memory leaks, but it does find the most common forms.

How to analyze a Java heap dump

- Single dump analysis
 - Inspects a single heap dump file for container objects that have large reach size as compared to their largest child object
 - Most commonly used to analyze automatically generated heap dumps after an OutOfMemoryException
- Comparative analysis (rarely necessary)
 - Analyzes heap growth between two dumps that are taken over time
 - Identifies the objects with the largest size difference and their ownership relationships
- Analysis results
 - Summary of analysis results that show heap contents, size, and growth
 - Lists suspected data structures, data types, and packages that contribute to the growth in heap usage
 - Tabular views of all the objects and data types in the memory dump with filters and sorted columns

© Copyright IBM Corporation 2013

Figure 8-59. How to analyze a Java heap dump

WA5913.0

Notes:

There are a few ways to go about analyzing heap dumps, and the best choice depends on what artifacts are available. If you have a single dump, likely generated on OOM, then single dump analysis is the only choice. However, if you have a dump from the same Java process (meaning before the OOM but after the most recent restart), you can use comparative analysis to better narrow down the leak suspects. In either case, most tools provide an analysis summary that breaks down the heap dump contents and aims to better direct your investigation.



IBM Monitoring and Diagnostic Tools for Java – Memory Analyzer Version 1.1

- Java heap analysis tool that is based on the Eclipse Memory Analyzer (MAT)
 - Available from IBM Support Assistant
- Memory Analyzer extends Eclipse MAT version 1.1 using the DTFJ
 - Enables Java heap analysis by using system core dumps and IBM Portable Heap Dumps (PHD)
- By using Memory Analyzer, you can:
 - Diagnose and resolve memory leaks that involve the Java heap
 - Derive architectural understanding of your Java application through footprint analysis
 - Improve application performance by tuning memory footprint and optimizing Java collections and Java cache usage
 - Produce analysis plug-ins with capabilities specific to your application
- Analyzes heapdumps for leaks and usage patterns
 - Identifies data structures that are potential leaks
 - Shows memory usage by Java package
 - Identifies areas for improvement to memory usage

© Copyright IBM Corporation 2013

Figure 8-60. IBM Monitoring and Diagnostic Tools for Java - Memory Analyzer Version 1.1

WA5913.0

Notes:

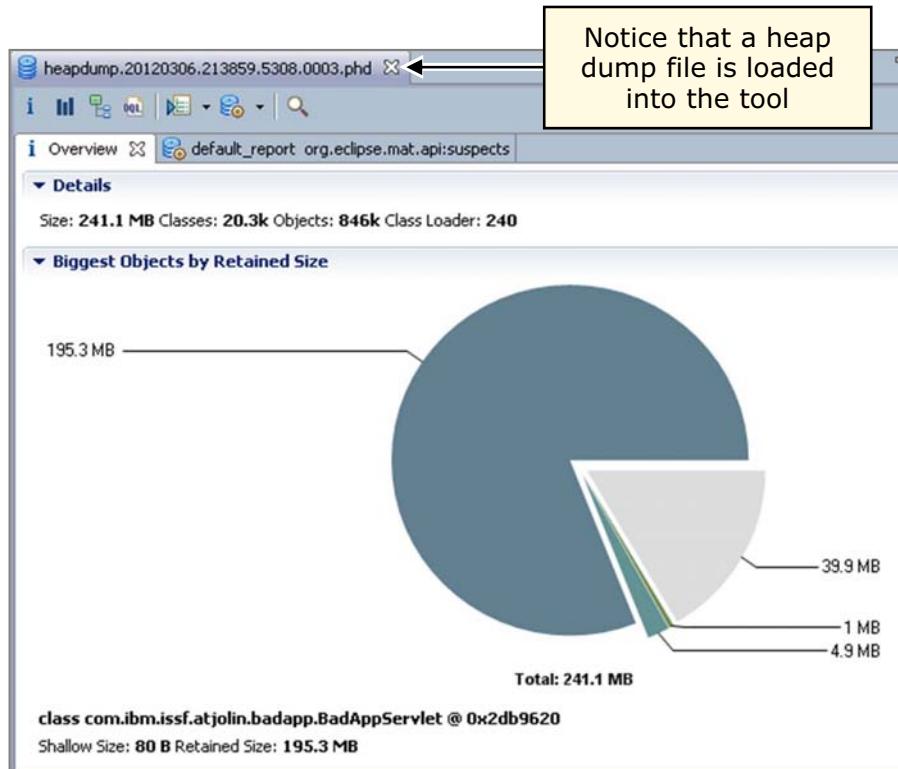
Memory Analyzer Tool is available from Eclipse.org: <http://www.eclipse.org/mat/>

DTFJ=Diagnostic Tool Framework for Java

WebSphere Education 

Memory Analyzer overview

- Pie chart of biggest objects and links to common analysis options
 - Histogram
 - Denominator tree
 - Top consumers
 - Duplicate classes



© Copyright IBM Corporation 2013

Figure 8-61. Memory Analyzer overview

WA5913.0

Notes:

This screen capture shows the Overview tab, which the Memory Analyzer displays after it analyzes a heap dump. This view presents a pie chart of the largest objects by retained size (reach size).

After an OutOfMemoryError was thrown, WebSphere generated the heap dump used for this analysis.

- Histogram: Lists number of instances per class
- Denominator Tree: Lists the biggest objects and what they keep alive
- Top Consumers: Prints the most expensive objects that are grouped according to class and package
- Duplicate Classes: Detects classes that are loaded by multiple class loaders

There is no exact algorithm for memory analysis. Use the overview, histogram, and dominator tree functions to do initial analysis. Use reports and queries for more detailed analysis.

When you open a heap dump file, the Overview tab provides a first analysis of the heap dump.

The general procedure is as follows:

1. Examine the overview.
2. Check the size of the heap dump, and the numbers of objects, classes, and class loaders that it contains.
3. Look at the biggest objects.
4. In the **Overview** tab, click the **Histogram** link. Use the Histogram view to group objects, examine or calculate the size of those objects, or run queries on those objects.
5. In the **Overview** tab, click the **Dominator Tree** link. Use the Dominator Tree view to find the largest objects, and to examine the dependencies between objects.



Memory Analyzer leak suspects: Default report

- One or more leak suspects are listed

The class "com.ibm.issf.atjolin.badapp.BadAppServlet", loaded by "com.ibm.ws.classloader.CompoundClassLoader @ 0x27fab88", occupies 204,815,600 (81.02%) bytes. The memory is accumulated in one instance of "java.lang.Object[]" loaded by "com.ibm.oti.vm.BootstrapClassLoader @ 0xaa2518".

Keywords

java.lang.Object[]
com.ibm.issf.atjolin.badapp.BadAppServlet
com.ibm.ws.classloader.CompoundClassLoader @ 0x27fab88
com.ibm.oti.vm.BootstrapClassLoader @ 0xaa2518

[Details >](#)

© Copyright IBM Corporation 2013

Figure 8-62. Memory Analyzer leak suspects: default report

WA5913.0

Notes:

The Leak Suspects view helps you to identify possible memory leak suspects. The report contains the same information as the heap dump overview report, and also a section about possible leak suspects. A summary pie chart shows the leak suspects and the proportion of the heap that each suspect is consuming.

Clicking the **Details** link reveals the call tree views for each of the suspects and more information such as:

- Shortest paths to the accumulation point
- Accumulated objects
- Accumulated objects by class

Denominator tree view

- MAT provides other views
 - The most useful for heap analysis is the Denominator Tree, which identifies the objects that hold the most heap

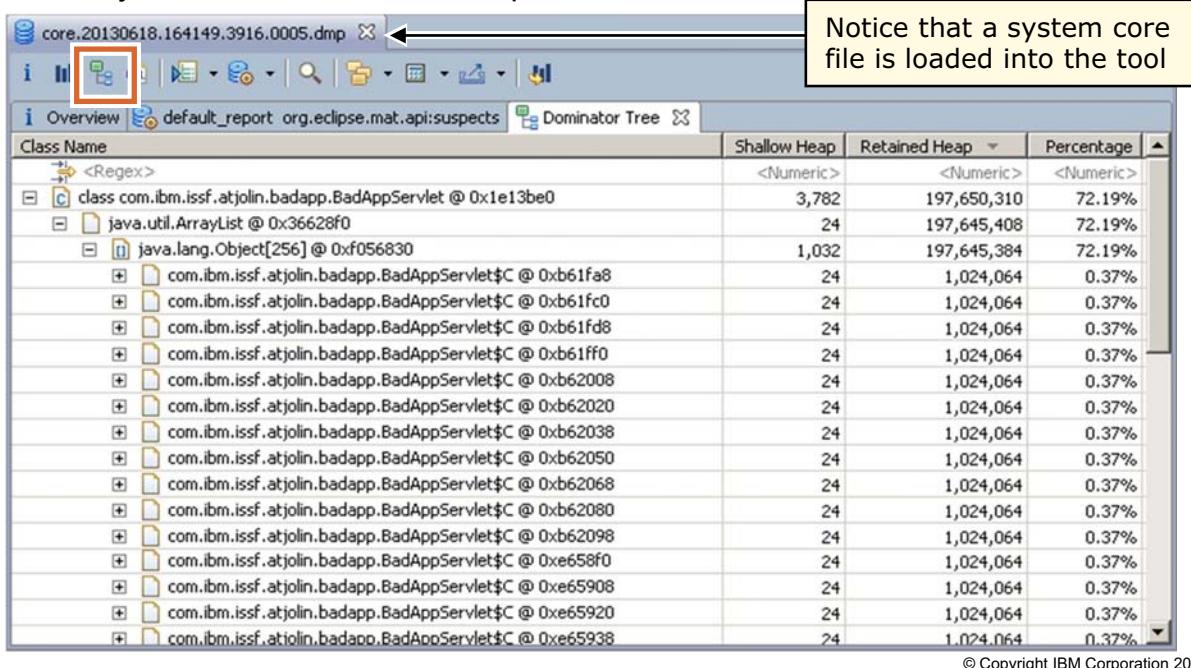


Figure 8-63. Denominator tree view

WA5913.0

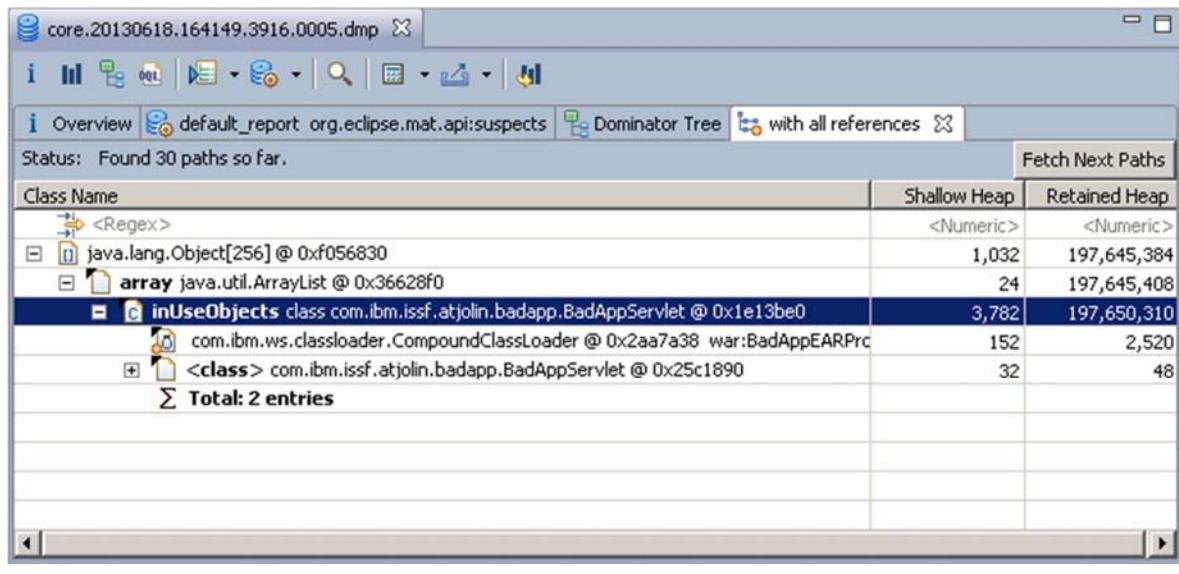
Notes:

To create the Denominator view, the object graph is converted to a tree with shared reference removed. The Shallow Heap column shows the amount of heap that is held by the object. The Retained Heap column shows the sum of the heap that is held by the object and all the descendants of that object. Focus on the objects at the top of the tree with the largest retained heap. Look for the “drop” in retained heap to signal the parent object as a heap suspect. In this example, the ArrayList object holds 72% of the heap. The array consists of hundreds of objects that use only a small amount of heap, 0.37%.

WebSphere Education

Path to GC roots

- After identifying the object that holds the most heap space alive, you can use the **path to GC roots** option to see what is keeping it alive:
 - Right-click the object to see the menu
 - You can ask to see all, or just the shortest path to the GC root



© Copyright IBM Corporation 2013

Figure 8-64. Path to GC roots

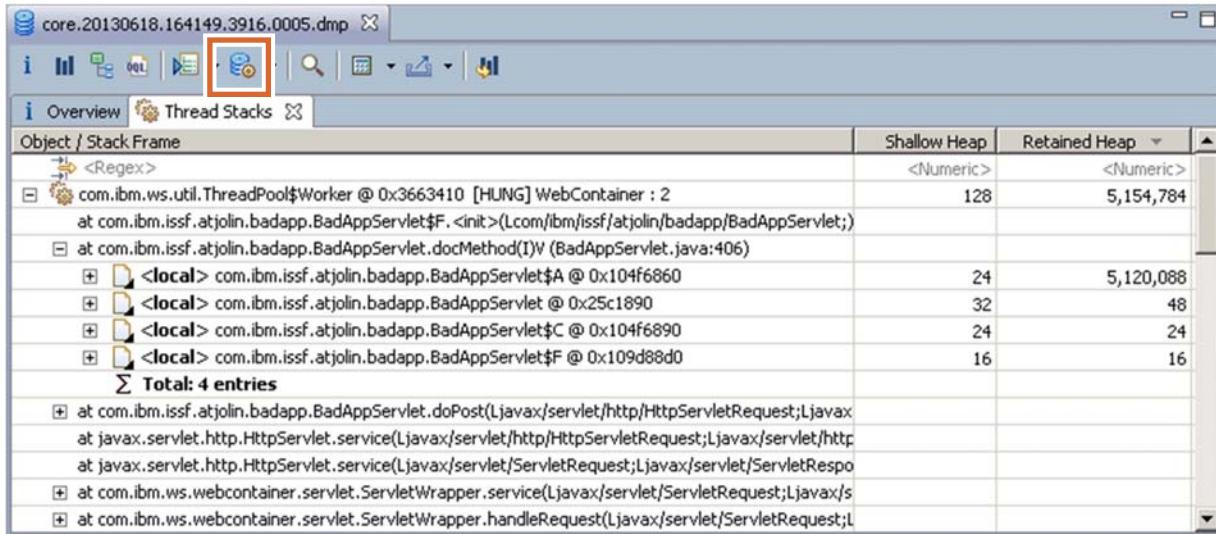
WA5913.0

Notes:

The object ArrayList is kept alive because there is still a reference to it in GC roots.

Thread stack view

- MAT also provides a Thread Stacks view that can be useful in understanding how a thread is influencing your heap usage
 - Accessed through the **Query browser > Java basics menu**



© Copyright IBM Corporation 2013

Figure 8-65. Thread stack view

WA5913.0

Notes:

The thread stack shows `<local>` variables that can be expanded to see their contents. Refer to the variable with the largest retained heap to signal a leak suspect.



IBM Extensions for Memory Analyzer

- The IBM Extensions for Memory Analyzer offer both additional capabilities for debugging generic Java applications, and capabilities for debugging specific IBM software products
- Extensions are currently available for:
 - Java SE runtime
 - WebSphere Application Server
 - CICS Transaction Gateway
- By using the IBM Extensions for Memory Analyzer you can:
 - Visualize the state of both your code and that of the IBM software products that are used in the application
 - Confirm the configuration of the IBM products by looking at the loaded configuration, and the state of the configured components: for example, cache and thread pool configurations
 - Inspect the size and contents of IBM product components, for example cache or HTTP session contents

© Copyright IBM Corporation 2013

Figure 8-66. IBM Extensions for Memory Analyzer

WA5913.0

Notes:

Java SE Runtime

- **Overview:** Provides a summary of the Java runtime and process, including vendor, version, operating system, Java home directory, class path, and command line.

WebSphere Application Server

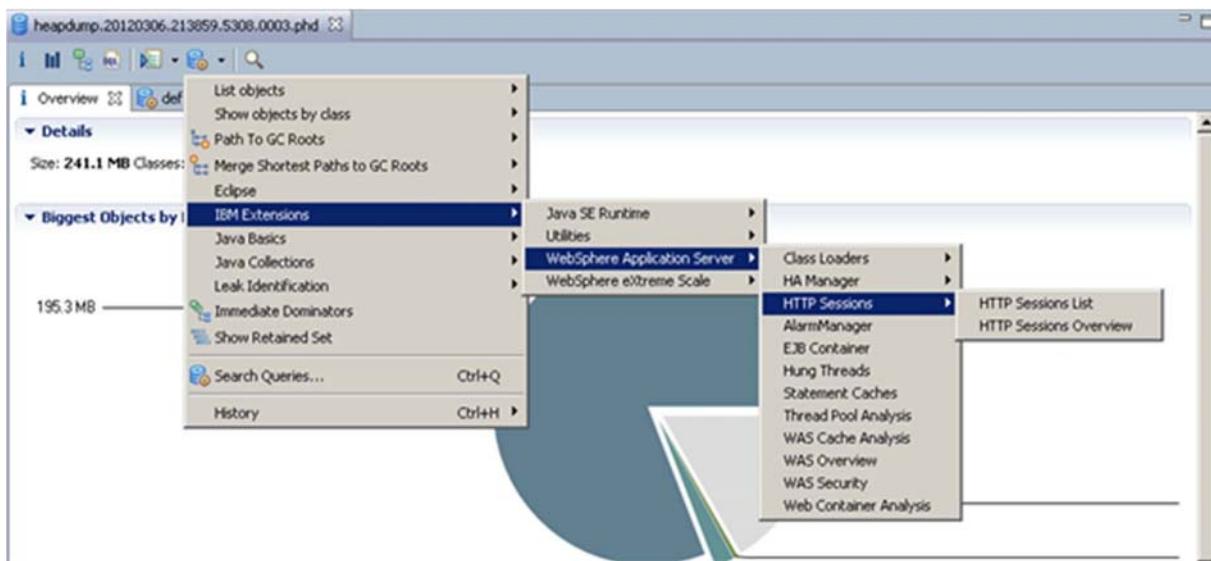
- **WebSphere Application Server Overview** provides a summary of the WebSphere Application Server runtime, including product and version, server name, node, cell, and the status of the deployed applications.
- **WebSphere Application Server Cache Analysis** provides a summary of the application server-provided caches, showing their size, occupancy, and retained memory.
- **Web Container Analysis** provides a breakdown of the deployed web applications, their locations, and the number of HTTP sessions that are associated with them.
- **Thread Pool Analysis** provides a table for the application server-provided thread pools that cover: thread pool configuration, current size, and presence of hung threads, plus a table for the Java level thread groups, covering thread count and amount of associated memory.

- **HTTP Sessions** produces a list of every HTTP session in memory, summarizing the application name, session ID, user name, timeout, plus creation and last access times. In addition, it is possible to expand any HTTP session to look at the keys and values that are associated with the session itself.
- **EJB Container** provides a simple breakdown of the number of each type of EJB deployed in the application server.
- **Application Classloader Leaks** lists all of the application class loaders, indicating whether any of them are stopped and therefore leaking. This information can then be used to determine which application is problematic.
- **Alarm Manager** provides a list of the deferrable, non-deferrable, deferred, and unqueued alarms.



IBM Extensions for Memory Analyzer

- The IBM Extensions can be accessed by clicking the “Disks” icon



© Copyright IBM Corporation 2013

Figure 8-67. IBM Extensions for Memory Analyzer

WA5913.0

Notes:

Utilities

- Find Garbage Fragments:** Searches the Java heap for fragments (chains) of objects on the Java heap that are eligible for garbage collection. These fragments are then displayed in descending order of retained memory.
- Export Object:** Enables an object graph to be written as a textual representation to a text file. The query allows you to use `-outputfile` to specify the output file and use `-maxdepth` to specify the depth of the tree to write.

WebSphere Application Server

- Overview:** The WebSphere Application Server Overview query is a simple query that shows the version and fix pack level of the WebSphere Application Server running. The query also shows the server, node, and cell name, the inferred startup time of the process, and a list of all applications and their running state. It also includes an experimental section that breaks down the memory usage of the Java heap according to components.

- **HTTP Sessions List:** One common class of problems is related to applications' usage of HTTP sessions. In addition to OutOfMemoryErrors, other issues include finding the contents of sessions, timeout values, session identifiers, and the applications to which the sessions belong. Many implementation details make finding sessions and their values nontrivial. These details include different class names across different versions of the product and multiple maps per session. (One is for persisted sessions and the other is for nonpersisted sessions, both of which might be active in a single session.) The HTTP Sessions List query extracts all of this information into a tree view.

HeapAnalyzer

- Analyzes heapdump for leaks and builds object hierarchy
 - Detection method is suited for large collections of objects
 - Simplifies hierarchy to one parent that might cause artifacts
- Available from IBM Support Assistant
- Available on alphaWorks:
 - <http://www.alphaworks.ibm.com/tech/heapanalyzer>
- Suited for quick and simple hierarchy inspection
- Some features include:
 - List of Java heap leak suspects
 - List of gaps among allocated objects/classes/arrays
 - Java objects, classes, and arrays search engine
 - List of objects, classes, and arrays by type, object, size, size of child, and other parameters
 - List of available heap spaces by size
 - Tree of Java heap dump
- Suggestion: Memory Analyzer is your first choice, use HeapAnalyzer only if you have some particular reason to prefer it

© Copyright IBM Corporation 2013

Figure 8-68. HeapAnalyzer

WA5913.0

Notes:

The final heap analysis tool that is mentioned is the HeapAnalyzer from alphaWorks. This tool does “in process” heap analysis and is suited for quick and simple hierarchy inspection, and detecting leaks due to large collections of objects. An in-depth tutorial walks through using HeapAnalyzer to detect a leak in a heap dump.

The tutorial is available at: <http://www.ibm.com/support/docview.wss?uid=swg27006624>

HeapAnalyzer summary

- Analysis of heap usage and general information

Property	Value
Heap dump file name	/Users/wareiche/Documents/WebSphere 7 PD Course/heapdump...
Java Version	J2RE 5.0 IBM J9 2.3 Windows XP x86-32 build 20080809_2189...
Number of Classes	12,799
Number of Objects	280,501
Number of ObjectArrays	43,025
Number of PrimitiveArrays	74,222
Total Number of Instances	410,547
Total Number of References	1,239,033
Number of roots	4,913
Number of types	12,799
Heap range	0x93e60 to 0x187938d0
Java heap usage	20,525,272 bytes
Dark Matter	2,526,016 bytes (12.306858 %)
WARNING!!	Java garbage collection trace may report 23,051,288 bytes are u... Actual Java heap usage is 20,525,272 bytes due to excessive dar...

© Copyright IBM Corporation 2013

Figure 8-69. HeapAnalyzer summary

WA5913.0

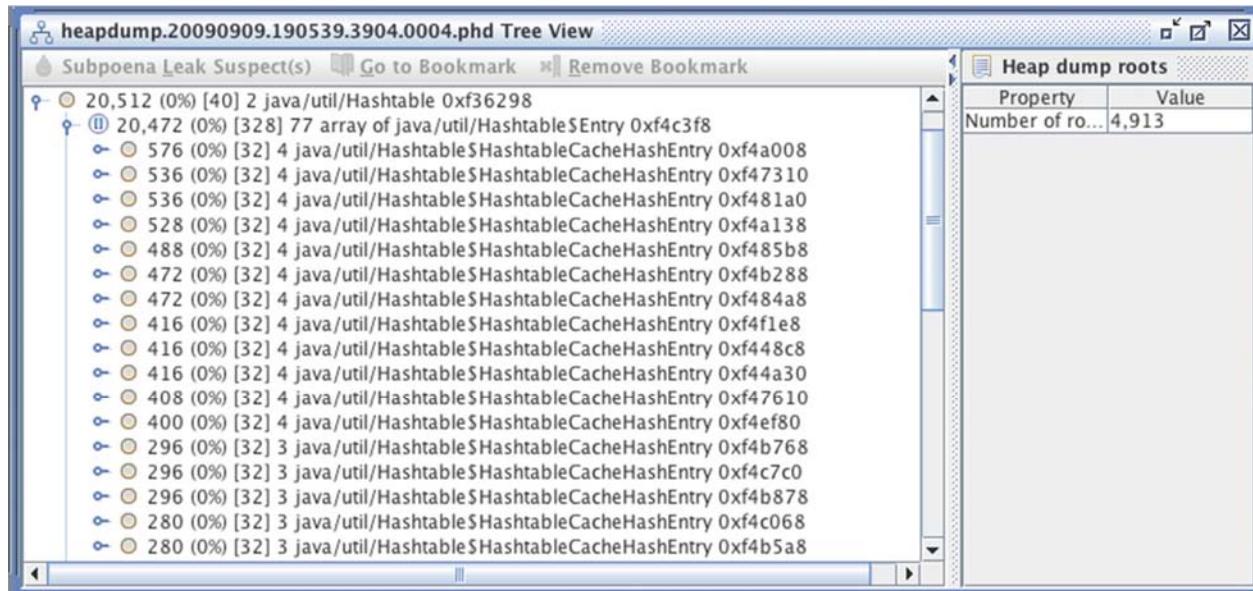
Notes:

The first view after HeapAnalyzer loads a heap dump is useful in determining the basic statistics for overall heap usage and often includes references to any findings from the built-in leak detection.



HeapAnalyzer tree

- Tree-based exploration of simplified object hierarchy can help identify leak suspects



© Copyright IBM Corporation 2013

Figure 8-70. HeapAnalyzer tree

WA5913.0

Notes:

HeapAnalyzer also includes a tree-based view that is based on a simplified object hierarchy. This view makes personal heap exploration easy to do, though the fact that it does hierarchy simplification means it might produce artifacts that incorrectly represent the hierarchy.

Unit summary

Having completed this unit, you should be able to:

- Define out-of-memory conditions
- Use monitoring tools to detect out-of-memory conditions
- Obtain and interpret a verbose GC log by using GCMV
- Obtain and analyze heap dumps and system core dumps
- Describe tools for analyzing out-of-memory problems

© Copyright IBM Corporation 2013

Figure 8-71. Unit summary

WA5913.0

Notes:

Checkpoint questions

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?

© Copyright IBM Corporation 2013

Figure 8-72. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. The following are three causes of Java heap OutOfMemory error:
 - Insufficient maximum heap size for user load
 - Memory leak in the Java code
 - Memory leak in native code
2. The following statistics can be monitored with the PMI Basic setting:
 - JVM UpTime
 - JVM HeapSize
 - JVM UsedMemory
3. The following diagnostic data is provided:
 - Size of object that is allocated
 - Available memory on the Java heap
 - JVM response to an allocation request

© Copyright IBM Corporation 2013

Figure 8-73. Checkpoint answers

WA5913.0

Notes:

Exercise 6



Troubleshooting an out-of-memory condition

© Copyright IBM Corporation 2013

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

8.0

Figure 8-74. Exercise 6

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Use Tivoli Performance Viewer to detect out-of-memory conditions
- Configure the lightweight memory leak detection mechanism
- Obtain verbose GC data and interpret it by using the Garbage Collection and Memory Visualizer tool (GCMV)
- Trigger a heap dump and analyze it by using the Memory Analyzer tool
- Examine the diagnostic data artifacts of an application that has a memory leak

© Copyright IBM Corporation 2013

Figure 8-75. Exercise objectives

WA5913.0

Notes:

Unit 9. Introduction to database connection problems

What this unit is about

This unit investigates the common problems that can arise when creating a connection to a database and the techniques that are used to solve them.

What you should be able to do

After completing this unit, you should be able to:

- Test data source connections to databases
- Describe Java EE Connector (J2C) authentication issues
- Describe Java database connectivity (JDBC) driver issues
- Use the built-in tester console to test connections
- Configure the JDBC driver trace facility

How you will check your progress

- Checkpoint questions
- Lab exercise

References

WebSphere Application Server Network Deployment, Version 6.1 Information Center, “Configuring a JDBC provider and data source”:
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/tdat_tccrtprovds.html

Unit objectives

After completing this unit, you should be able to:

- Test data source connections to databases
- Describe Java EE Connector (J2C) authentication issues
- Describe Java database connectivity (JDBC) driver issues
- Use the built-in tester console to test connections
- Configure the JDBC driver trace facility

© Copyright IBM Corporation 2013

Figure 9-1. Unit objectives

WA5913.0

Notes:

WebSphere Application Server V7.0 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 V7.0 as a JDBC provider. At least one data source is associated with each JDBC provider and can be configured through the WebSphere Application Server V7.0 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. Before deploying applications that use the database connection, the data source can be verified by using the *test connection* service of WebSphere Application Server V7.0.

JDBC providers and JDBC data sources in WebSphere Application Server V7.0 have few changes 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 JDBC providers are based on the Derby JDBC classes.



Topics

- Overview of database connection concepts
- Common database connection problems

© Copyright IBM Corporation 2013

Figure 9-2. Topics

WA5913.0

Notes:

9.1. Overview of database connection concepts

Overview of database connection concepts



© Copyright IBM Corporation 2013

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

8.0

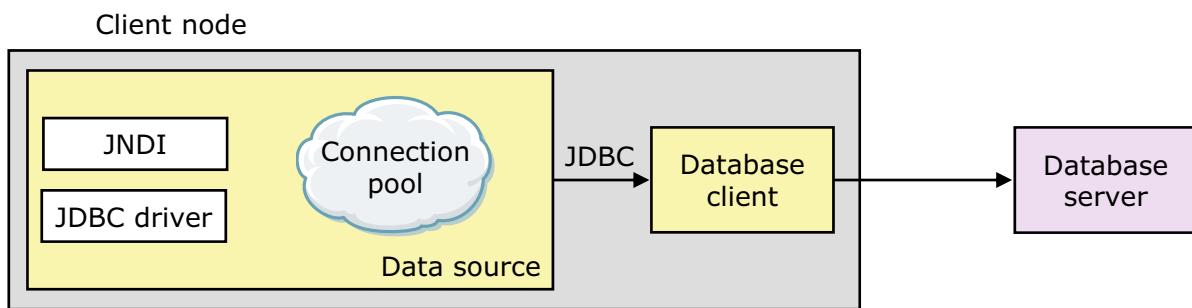
Figure 9-3. Overview of database connection concepts

WA5913.0

Notes:

JDBC providers

- Provide the JDBC driver implementation for database access
 - Type 2 JDBC drivers (thick) - Require the database client software on the client node to connect to the database server
 - Type 3 JDBC drivers (net protocol) - Require server-side code to map net protocol to the native database
 - Type 4 JDBC drivers (native protocol) - Connect directly to the database by using its native protocol
- XA drivers support transaction recovery



© Copyright IBM Corporation 2013

Figure 9-4. JDBC providers

WA5913.0

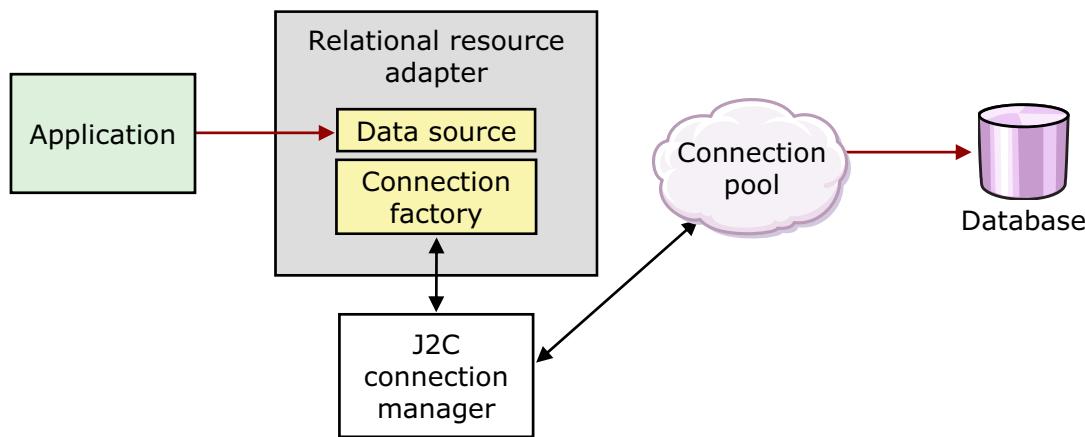
Notes:

The JDBC provider object supplies the specific JDBC driver implementation class for access to specific vendor database. The driver implementation classes are supplied by defining the WebSphere variables that are associated with the provider. Drivers fall into one of the following categories: The type 2 JDBC driver or thick driver requires the database client software on the client node to connect to the database server. The type 3 JDBC driver requires server-side code to map net protocol to the native database. The type 4 JDBC driver, or thin driver, uses its native protocol to connect directly to the database. This information is specified as part of the data source configuration.

z/OS note: In addition to JDBC drivers, zWSAS can use CTG CICSECIConnector and IMS Connect drivers to connect to CICS and IMS.

Data sources

- Data sources can improve performance and portability for database access
 - Standard and XA data sources
- Connection pooling requires two parts:
 - J2C connection manager
 - Relational resource adapter



© Copyright IBM Corporation 2013

Figure 9-5. Data sources

WA5913.0

Notes:

Rather than having the JDBC drivers directly communicate with the database, the communication is abstracted into a data source.

WebSphere Application Server provides the WebSphere Relational Resource Adapter implementation. 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 installed and runs as part of WebSphere Application Server, and needs no further administration.

The RRA supports both the configuration and use of JDBC data sources and Java EE Connector Architecture (JCA) connection factories. The RRA supports the configuration and use of sources that are implemented as either JDBC data sources or Java EE Connector Architecture connection factories. Applications can use data sources directly, or container-managed persistence (CMP) entity beans can configure data sources for use.

Enhanced EAR

- Enterprise archive that contains Java EE artifacts plus resource information that is needed to install on WebSphere Application Server
 - JDBC resources (data sources)
 - Classloader
 - JAAS authentication aliases
 - Shared libraries
 - Virtual host information
- Support integrated with the IBM Assembly and Deploy Tools (IADT)
 - Found on the Deployment page of the application deployment descriptor
- **Warning:** Can possibly cause problems if unintended application scoped resources are used in production
 - Enhancements can be removed or ignored during application installation

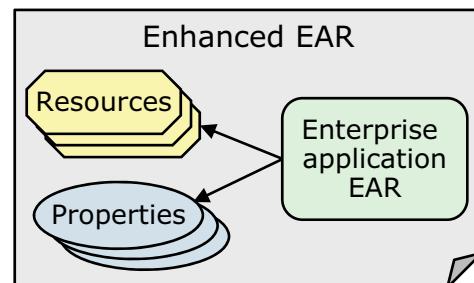


Figure 9-6. Enhanced EAR

© Copyright IBM Corporation 2013

WA5913.0

Notes:

The enhanced EAR is IBM WebSphere specific. It is not part of Java EE. However, those artifacts that are defined within an enhanced EAR are ignored if this EAR file is installed on an application server that is not WebSphere.

Developers or administrators can define resources and properties within an enterprise application in tools and import or export the enhanced EAR file.

Some resources must still be defined in the Application Server, such as JMS and JavaMail. Settings are defined in IBM tools, which are stored on `deployment.xml`, and packaged with the EAR file. Resources are applied at the new application scope.

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 that are associated with the provider
- Environment variables are used to specify Java classes and native libraries:
 - DB2UNIVERSAL_JDBC_DRIVER_PATH
 - INFORMIX_JDBC_DRIVER_PATH
 - DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH
 - ORACLE_JDBC_DRIVER_PATH
- Custom variables can be defined in WebSphere Application Server to accommodate any vendor-specific driver implementation

© Copyright IBM Corporation 2013

Figure 9-7. The role of the JDBC provider

WA5913.0

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 to function. Therefore, it is necessary to reference one or more native libraries. JDBC type 3 and 4 drivers are pure Java drivers, and they require only 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. To configure this driver, the DB2_UNIVERSAL_DRIVER_PATH and UNIVERSAL_DRIVER_PATH WebSphere variables 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 is 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`.

JDBC provider configuration

The screenshot shows the JDBC provider configuration screen in the WebSphere Application Server V7.0 administrative console. The configuration panel has tabs for 'General Properties' and 'Advanced Properties'. The 'General Properties' tab is selected. The following fields are visible:

- 1 Scope:** cells:was7host01Cell01:nodes:was7host01Node01:servers:server1
- 2 Name:** DB2 Universal JDBC Driver Provider (XA)
- 3 Class path:** \${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar
\${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar
\${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar
- 4 Native library path:** \${DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH}
- 5 Isolate this resource provider:**
- 6 Implementation class name:** com.ibm.db2.jcc.DB2XADDataSource

1. Scope
2. Name
3. Class path (Java)
4. Native path (C code)
5. Isolate provider (new V7)
6. Implementation class name

© Copyright IBM Corporation 2013

Figure 9-8. JDBC provider configuration

WA5913.0

Notes:

The JDBC provider screen can be accessed through the WebSphere Application Server V7.0 administrative console by navigating to **Resources > JDBC > JDBC Providers > JDBC Provider name**. You use this panel to specify the name, description, any JAR file references, any native library references, and the implementation class name for the provider.

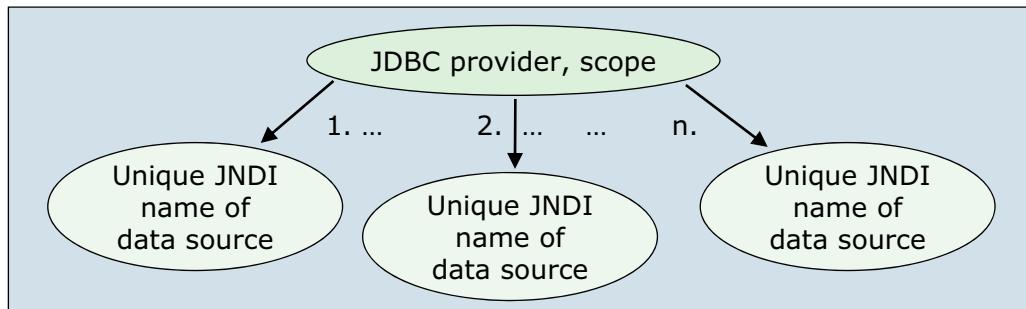
The properties of the JDBC provider screen include:

1. **Scope:** The namespace of the JDBC provider (cell, node, or server scope).
2. **Name:** The display name of this JDBC provider.
3. **Class path:** The Java CLASSPATH addition.
4. **Native path:** The system PATH addition.
5. **Isolate this resource provider (new in V7):** Specifies that this resource provider is loaded in its own class loader. This specification allows different versions or implementations of the same resource provider to be loaded in the same Java virtual machine. Give each version of the resource provider a unique class path that is appropriate for that version or implementation.

6. **Implementation class file name:** The vendor-specific Java class that implements the JDBC provider.

The role of the data source

- The JNDI name represents the runtime binding of the data source to the JDBC provider in a specific scope
 - A data source is analogous to a Java EE Connector Architecture (JCA) connection factory



- Runtime binding creates a highly flexible environment but if problems exist, the symptoms are not seen until run time
 - The test connection facility of the administrative console is helpful for uncovering such errors during the deployment phase

© Copyright IBM Corporation 2013

Figure 9-9. The role of the data source

WA5913.0

Notes:

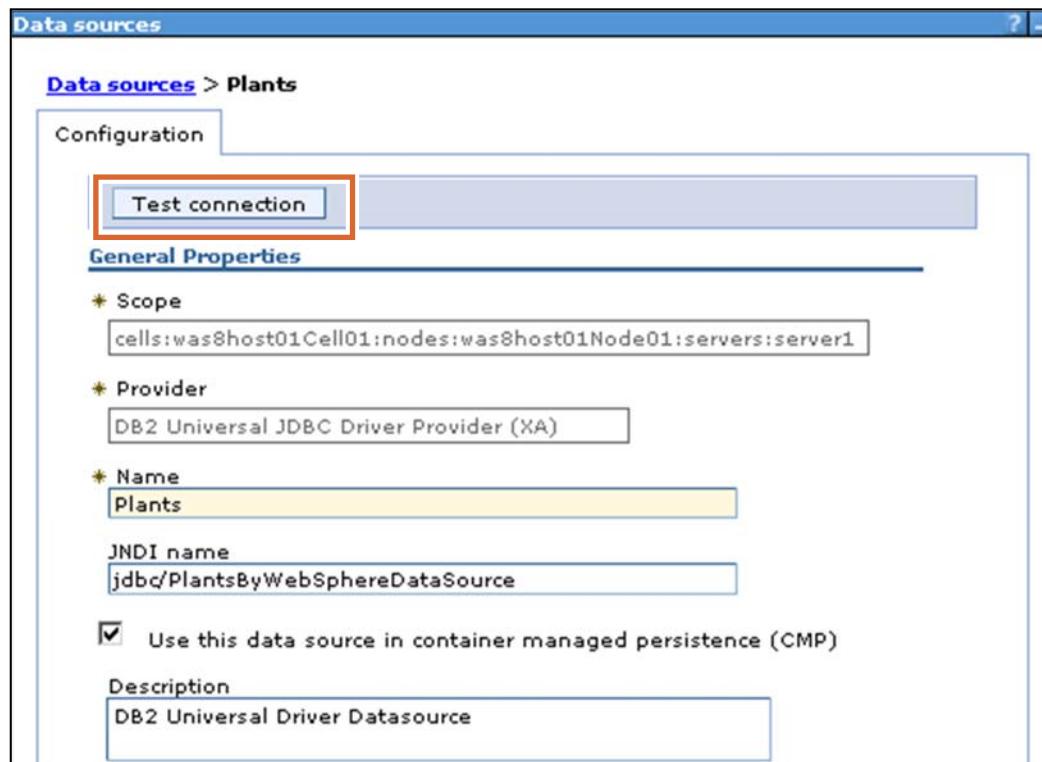
A data source requires information that includes:

- The server name
- The listening port of the database server
- The user name credentials
- The type of driver that is used for the connection

The data source is bound to the runtime environment through its JNDI name. A JDBC provider and the scope have a one-to-many relationship with the JNDI name of the data source. In WebSphere Application Server V7.0, 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, including: the database name, the database server name, the type of the driver that is used, the database port number, and the authentication credentials that determine authorities and privileges for the connection. The authentication credentials are specified by using a Java 2 Connector 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 by using a JNDI name. This relationship 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 are not detected until the references are exercised and a JNDI resolution is attempted.

Data source screen (1 of 3)



© Copyright IBM Corporation 2013

Figure 9-10. Data source screen (1 of 3)

WA5913.0

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 the application uses 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).



Data source screen (2 of 3)

Data store helper class name

Select a data store helper class

Data store helper classes provided by WebSphere Application Server

DB2 Universal data store helper
`(com.ibm.websphere.rssadapter.DB2UniversalDataStoreHelper)`

Specify a user-defined data store helper

Enter a package-qualified data store helper class name

Security settings

Select the authentication values for this resource.

Authentication alias for XA recovery

Component-managed authentication alias

Mapping-configuration alias

Container-managed authentication alias

Set authentication alias if your database requires the user ID and password to get a connection; avoid a runtime exception

© Copyright IBM Corporation 2013

Figure 9-11. Data source screen (2 of 3)

WA5913.0

Notes:

Data store helper class name: Specifies the name of the DataStoreHelper implementation class that extends the capabilities of your selected JDBC driver implementation class to perform database-specific functions.

The application server provides a set of DataStoreHelper implementation classes for each of the JDBC provider drivers that it supports. These implementation classes are in the package `com.ibm.websphere.rssadapter`. For example, if your JDBC provider is DB2, then your default DataStoreHelper class is `com.ibm.websphere.rssadapter.DB2DataStoreHelper`. The administrative console page that you are viewing, however, might make multiple DataStoreHelper class names available to you in a list; be sure to select the one required by your database configuration. Otherwise, your application might not work correctly. If you want to use a DataStoreHelper class that is not listed in the list, select **Specify a user-defined DataStoreHelper**, and type a fully qualified class name. See the information center for instructions on creating a custom DataStoreHelper class.

Authentication alias for XA recovery: This field is used to specify the authentication alias that you must use during XA recovery processing. If this alias name is changed after a server failure, the subsequent XA recovery processing uses the original setting that was in effect before the failure.

Component-managed authentication alias: This alias is used for database authentication at run time.

If your database is not secured, setting database authentication is not required. Having an unsecured database is not a good practice for a production environment.

Avoid trouble: If you have a database that does not support user ID and password, like Cloudscape, do not set the alias in the component-managed authentication alias or container-managed authentication alias fields. Otherwise, you see the warning message in the system log to indicate that the user and password are not valid properties. This message is only a warning message; the data source is still created successfully.

If you do not set an alias through the component-managed authentication or otherwise, and your database requires the user ID and password to get a connection, an exception occurs during run time.



Data source screen (3 of 3)

Common and required data source properties

Name	Value
* Driver type	4
* Database name	PLANTS
* Server name	dbhost
* Port number	50000

- DNS must resolve the host name of the database server correctly
- The port number must be a valid port on the database server

© Copyright IBM Corporation 2013

Figure 9-12. Data source screen (3 of 3)

WA5913.0

Notes:

This portion of the data source screen shows the configuration for the database. The database name must be correct for the database that exists on the database server. The server name is the host name of the database server, which must be resolved correctly with DNS. The port number must be a valid port on the database server.

9.2. Common database connection problems

Common database connection problems



© Copyright IBM Corporation 2013

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

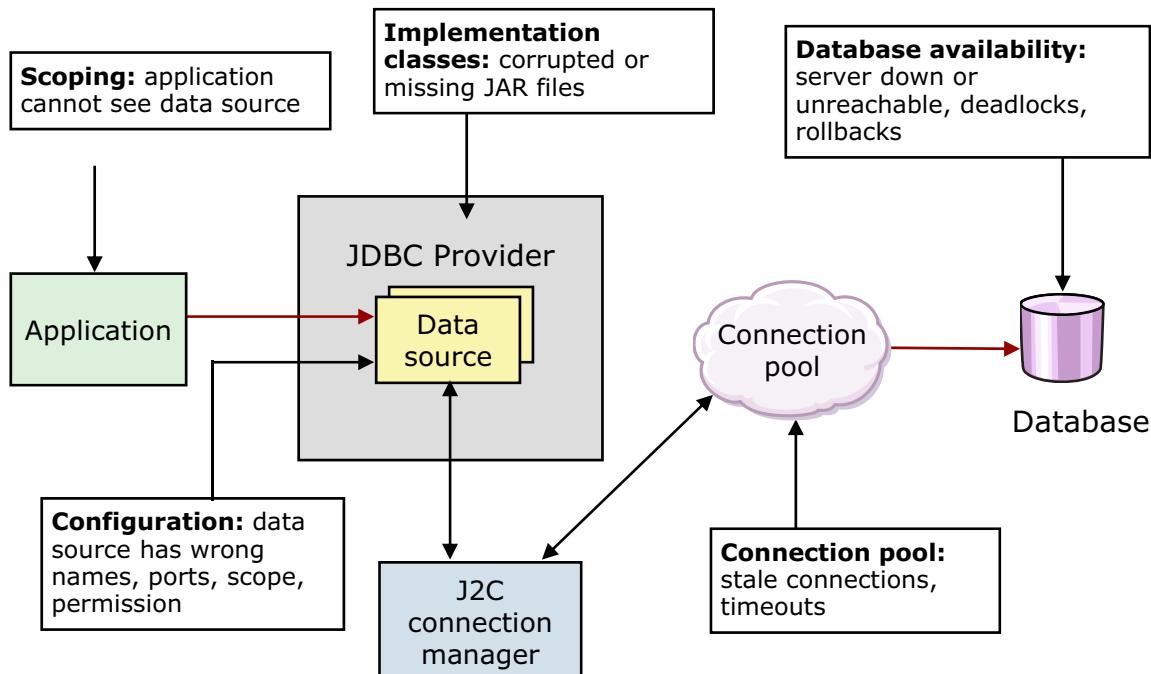
8.0

Figure 9-13. Common database connection problems

WA5913.0

Notes:

What can go wrong



© Copyright IBM Corporation 2013

Figure 9-14. What can go wrong

WA5913.0

Notes:

Typical causes of database connection problems include:

- **Scoping:** Application cannot see data source
- **Implementation classes:** Corrupted or missing JAR files
- **Database availability:** Server down or unreachable
- **Configuration:** Data source has wrong names, ports, scope, permission
- **Connection pool:** Stale connections, timeouts

Creating a database connection: Common problems

- Typical problems that arise when configuring a database connection are:
 - Spelling or typographical errors when entering 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 is not started
- The problems that are encountered when configuring a database connection are usually easy to fix if you know what to look for and where to look
 - WebSphere provides immediate feedback when testing a connection
 - The `SystemErr.log` contains explicit information about the actual exception

© Copyright IBM Corporation 2013

Figure 9-15. Creating a database connection: Common problems

WA5913.0

Notes:

Spelling any of the database connection properties incorrectly, or violating any of the case sensitivity rules of a parameter, results in failure when verifying the connection with the *test connection* service. Incorrectly creating a WebSphere variable or neglecting to make the necessary JAR file or native libraries available to the connection also leads 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 by using the server name value that you are using for the data source.
2. Start a database session by using the JCA credentials.
3. Perform a sample of the operations that the application performs, such as a simple query.
4. Disconnect from the machine.

If you are successful in 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 from where you are running the Telnet test, host resolution issues can occur. Also, the Telnet test does not exercise the port number parameter. If this port number is incorrect, then the data source fails. You can verify the port number that the database is listening on by exporting the instance parameters. This action assumes that you have the authority to do so. If not, you need to contact your database administrator and ask to confirm the parameters.



Using the test connection service (1 of 2)

- After the JDBC Provider and the data source are configured, select **JDBC Providers > data_source_name > Data sources** and click **Test connection**
- Test connection** is also available on the actual data source configuration panel of the administrative console

You can administer the following resources:					
Select	Name	JNDI name	Scope	Provider	Description
<input type="checkbox"/>	Default Datasource	DefaultDatasource	Node=was8host01Node01,Server=server1	Derby JDBC Provider	Datasource for the WebSphere Default Application
<input checked="" type="checkbox"/>	Plants	jdbc/PlantsByWebSphereDataSource	Node=was8host01Node01,Server=server1	DB2 Universal JDBC Driver Provider (XA)	DB2 Universal Driver Datasource
Total 2					

© Copyright IBM Corporation 2013

Figure 9-16. Using the test connection service (1 of 2)

WA5913.0

Notes:

The **Test connection** button is 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 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 are applied and saved to the application server configuration files.

Using the test connection service (2 of 2)

- If the data source is configured properly, you receive a message similar to the following example:



- If a failure message occurs, then it is time to track down the cause of the failure and resolve it
- A failure might be caused for a number of reasons, and is the subject of the next slides

© Copyright IBM Corporation 2013

Figure 9-17. Using the test connection service (2 of 2)

WA5913.0

Notes:

WebSphere Application Server Version V7.0 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 to track down the root cause of the issue.

JDBC provider configuration problems: Other considerations

- If you are using a deployment manager scenario, verify that the variables are defined in terms of other variables, such as WAS_INSTALL_ROOT for the application server installation path
- **Important:** Make sure that the scope of each variable you are defining is visible to the variables used in the JDBC provider
 - Whether they are cell, node, or server, scope might be significant

© Copyright IBM Corporation 2013

Figure 9-18. JDBC provider configuration problems: Other considerations

WA5913.0

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 files and native libraries that the JDBC provider requires can be local, 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 important to the visibility of each component. Defining the driver implementation at the server level leads 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 result in class loader exceptions because the system cannot locate the necessary JAR files or native libraries.

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

© Copyright IBM Corporation 2013

Figure 9-19. Data source database parameter problems: Identification

WA5913.0

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 menu. Exception events can be viewed separately, or a portion of the log files can be viewed. Because the log files can become 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 that contains the failure 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.

Data source database parameter problems: Diagnosis and resolution

- The `SystemErr.log` file tells you that the address provided is invalid for the connection
- A connectivity problem might be the result of an incorrect database name, server name, or port number
 - The database administrator must be consulted to verify the database name
 - You can verify that the server name is correct by attempting a ping of the host name or a Telnet session
 - The port number might be 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

```
SystemErr      R java.net.SocketException: Operation timed out:  
connect:could be due to invalid address      at  
java.net.PlainSocketImpl.socketConnect(Native Method)
```

© Copyright IBM Corporation 2013

Figure 9-20. Data source database parameter problems: Diagnosis and resolution

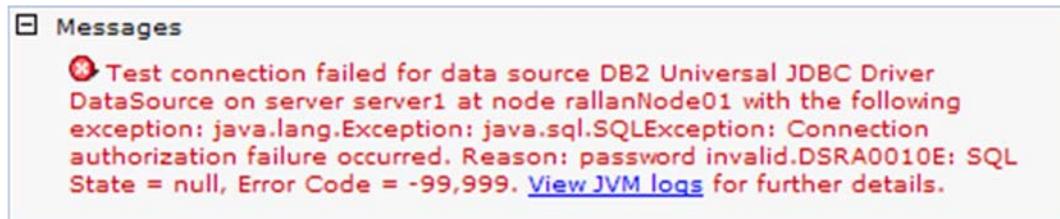
WA5913.0

Notes:

Determining the database name, server name, and port number varies 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 is something that the developers can discover for themselves. In a production environment, however, dedicated administrators or even separate groups commonly control machine access. Therefore, it might be necessary to engage the database administrator to resolve the connectivity issue.

Data source database authentication problems: Identification

- If you run the **test connection** service and receive an error message



- It is signifying a problem with the authentication credentials
- This issue might 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

© Copyright IBM Corporation 2013

Figure 9-21. Data source database authentication problems: Identification

WA5913.0

Notes:

Before WebSphere Application Server Version 5, data source configuration also included the user name and password information. Beginning with WebSphere Application Server Version 5, Java EE Connector Architecture (JCA) was introduced; 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 7.0 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 that are associated with the *version 4 data source* need to be verified. If you have many of these legacy data sources and the credentials are incorrect, then they must be fixed in each data source configuration. One of the advantages of using a JCA-compliant authentication scheme is that when you fix only one set of credentials, all of the data sources pick up the correction.

Data source database authentication problems: Diagnosis and resolution

- The exception in the `SystemErr.log` file says that the password is invalid:

```
SystemErr      R java.lang.Exception: java.sql.SQLException: Connection
authorization failure occurred. Reason: password invalid.DSRA0010E: SQL
State = null, Error Code = -99,999
```

- In this particular case, it was as a result of assigning the incorrect JCA authentication definition to the data source
 - Inspect all parameters that are associated with the user authentication and verify that they are correct
- Coordinate with the database administrator or server security administrator
 - Verify that the user account actually exists and is a member of the group that is able to access the database

© Copyright IBM Corporation 2013

Figure 9-22. Data source database authentication problems: Diagnosis and resolution

WA5913.0

Notes:

Verify that user credentials include activities such as logging in to the database server and verifying access to the machine. Depending on the database, security services can be provided from the local operating system, LDAP server, or some other form of access control. In a larger organization, verifying the user credentials can involve several people and cross over several groups. As was explained previously, a simple Telnet test to the database server might be necessary to verify that the credentials are correct and have the necessary authorities that are required for the data source.

Database deadlock problems

- This problem is probably an application-caused DB2 deadlock, particularly if you see an error similar to the following when accessing a DB2 data source:

```
ERROR CODE: -911 COM.ibm.db2.jdbc.DB2Exception: [IBM] [CLI Driver] [DB2/NT]
SQL0911N The current transaction has been rolled back because of a
deadlock or timeout. Reason code "2". SQLSTATE=40001
```

- To diagnose the problem, run these DB2 commands:
 - db2 update monitor switches using LOCK ON
 - db2 get snapshot for LOCKS on <dbName >
- Turn off the lock monitor by running:
 - db2 update monitor switches using LOCK OFF

© Copyright IBM Corporation 2013

Figure 9-23. Database deadlock problems

WA5913.0

Notes:

The `lock_snapshot.log` file contains the DB2 lock information.

To verify that you have a deadlock:

- Look for an application handle that has a lock-wait status, and then look for the ID of the agent that is holding the lock to verify the ID of the agent.
- Go to that handle to verify that it has a lock-wait status, and the ID of the agent that is holding the lock for it. If it is the same agent ID as the previous one, then you know that you have a circular lock (deadlock).

To resolve the problem, see the information center topic: *Data access problems for DB2 databases*:
http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Frtrb_dsaccess3.html

Database lock contention and rollback problems

- Symptom: A lock contention exception occurs in a DB2 database that your application accesses through an XA data source
- Problem
 - Your application is trying to access database records that the XA transaction locked and that are in ended (e) state, but the transaction manager cannot prepare
 - Because it is not considered to be in doubt, the transaction manager cannot recover this transaction
 - DB2 does not return it in the list of in doubt transactions
- DB2 does not roll back the transaction immediately
 - Waits until all connections to the database are released
 - During this period of inaction, the transaction continues to hold locks on the database
 - Application server does not disconnect all connections from the database to allow rollback, the ended transaction persists in locking the same database records
 - If your application attempts to access these locked records, a lock contention exception occurs in DB2

© Copyright IBM Corporation 2013

Figure 9-24. Database lock contention and rollback problems

WA5913.0

Notes:

Suggested response: DB2 Version 8.2 is packaged with a sample application that connects to a defined DB2 server and uses the available DB2 APIs to obtain a list of these particular ended transactions. The application offers a configuration setting that enables you to designate an amount of time after which the application rolls back these transactions. Locate the sample application in the `sqlib/samples/db2xamон.c` directory of DB2 Version 8.2, and run it.

Overview of the DB2 Universal Driver trace facility

- Many JDBC drivers have a trace or debug facility available
- The DB2 Universal Driver has extensive trace capabilities that can be activated from the 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` logs everything available to the trace file

© Copyright IBM Corporation 2013

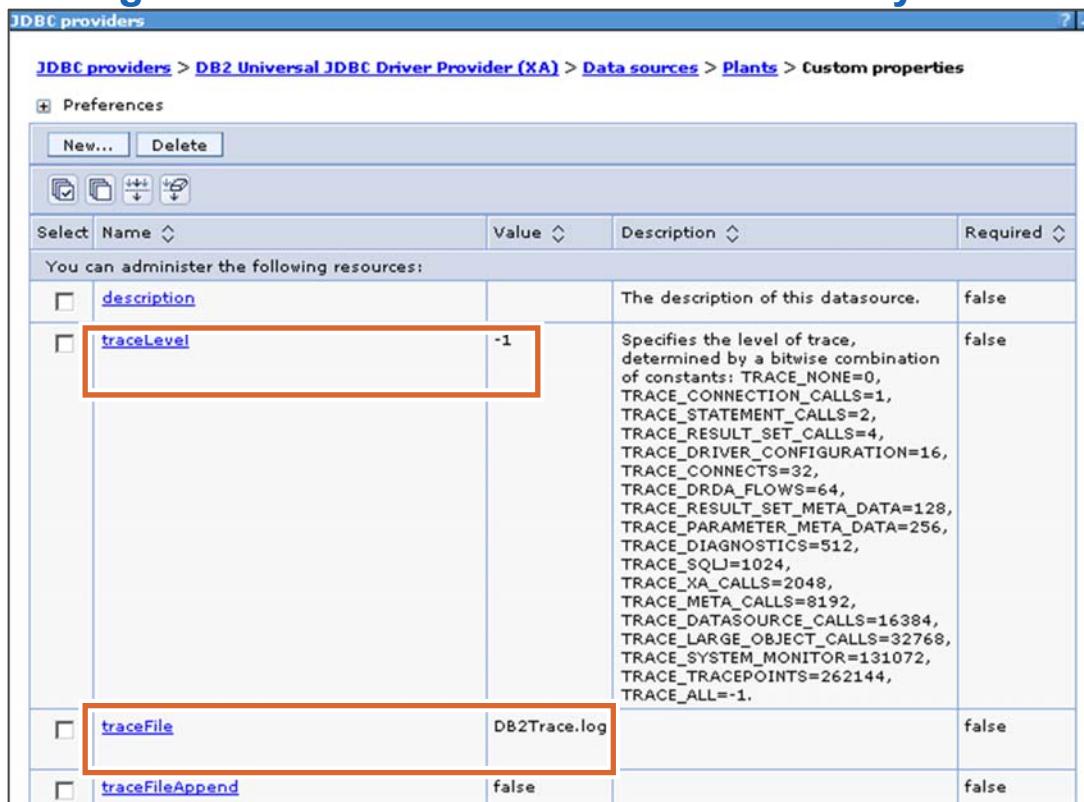
Figure 9-25. Overview of the DB2 Universal Driver trace facility

WA5913.0

Notes:

The trace settings, output, and interpretation vary from vendor to vendor. The **Custom properties** panel for a data source is populated with parameters that are specific to the driver implementation of the vendor. For JDBC providers and data sources that WebSphere Application Server Version 7.0 supports, custom properties are 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 need to refer to the documentation to activate driver tracing.

Enabling the DB2 Universal Driver trace facility



© Copyright IBM Corporation 2013

Figure 9-26. Enabling the DB2 Universal Driver trace facility

WA5913.0

Notes:

The DB2 Universal Driver Trace facility is 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 three parameters of interest are:

- **traceLevel**: In this example, it is set to -1, full trace.
- **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 is logged in to the WebSphere Application Server V7.0 trace file.
- **traceFileAppend**: Specifies whether to append to or overwrite the file that the `traceFile` property specifies. The default is false, which means that the file that the `traceFile` property specifies is overwritten.

DB2 Universal Driver trace facility output example

```
[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@658770921] 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)
```

© Copyright IBM Corporation 2013

Figure 9-27. DB2 Universal Driver trace facility output example

WA5913.0

Notes:

The trace file can grow large, quickly. Attempting to debug a driver problem in a production environment can affect performance and result in a trace file that is difficult to view and transfer. Obviously there are situations where only production systems expose some class of problems. These situations are probably less likely when resolving runtime reference issues, such as those associated with JDBC. Therefore, whenever possible, create a simple test that isolates the problem and run 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 by using the *test connection* facility. It shows the driver information and the exception that is logged as a result of an improperly configured database name in the data source properties.



Enabling Oracle JDBC driver tracing

The screenshot displays the "JDBC providers" administrative interface. Under "Oracle JDBC Driver (XA) > Data sources > [Data Source Name]", it shows a list of custom properties:

Select	Name	Value
<input type="checkbox"/>	driverType	thin
<input type="checkbox"/>	oracleLocFileSizeLimit	0
<input type="checkbox"/>	oracleLocFileCount	1
<input type="checkbox"/>	oracleLocFileName	
<input type="checkbox"/>	oracleLogTraceLevel	FINEST
<input type="checkbox"/>	oracleLogFormat	SimpleFormat
<input type="checkbox"/>	oracleLogPackageName	oracle.jdbc.driver

© Copyright IBM Corporation 2013

Figure 9-28. Enabling Oracle JDBC driver tracing

WA5913.0

Notes:

You can use the administrative console to enable Oracle JDBC driver tracing as well. The screen shows the Oracle log trace level that is set to finest.

Unit summary

Having completed this unit, you should be able to:

- Test data source connections to databases
- Describe Java EE Connector (J2C) authentication issues
- Describe Java database connectivity (JDBC) driver issues
- Use the built-in tester console to test connections
- Configure the JDBC driver trace facility

© Copyright IBM Corporation 2013

Figure 9-29. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. True or false: The only JDBC driver implementations that WebSphere Application Server supports are those drivers for which there are predefined 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 preconfigured for the common JDBC providers.
4. Name two common sources of JDBC provider configuration error.

© Copyright IBM Corporation 2013

Figure 9-30. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

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

Checkpoint answers

1. False. WebSphere supports any JDBC-compliant database. New variables can be defined to describe the driver implementation in the environment.
2. False. The data source is analogous to a JCA connection factory.
3. False. Environment variables must be configured for some driver implementations. Common drivers are preconfigured.
4. An incorrectly specified environment variable value, and a missing JAR file or library file.

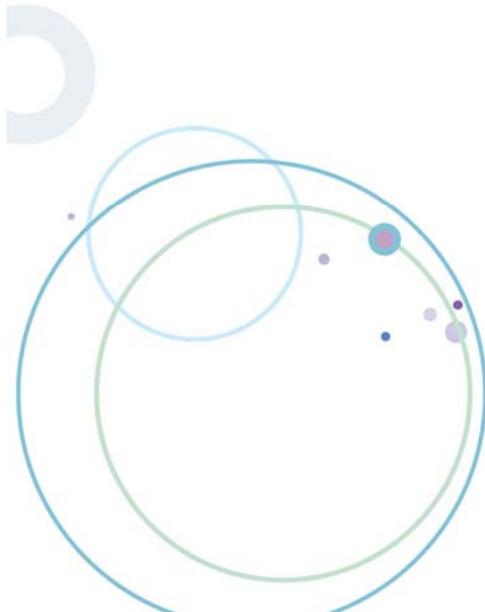
© Copyright IBM Corporation 2013

Figure 9-31. Checkpoint answers

WA5913.0

Notes:

Exercise 7



Troubleshooting database connection problems

© Copyright IBM Corporation 2013

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

8.0

Figure 9-32. Exercise 7

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Access and interpret runtime messages in the administrative console
- Test a data source connection to a database by using the Test connection facility in the administrative console
- Examine server logs by using the HPEL Log Viewer
- Locate and examine FFDC logs
- Configure tracing for a data source and interpret the trace data
- Troubleshoot database connection problems

© Copyright IBM Corporation 2013

Figure 9-33. Exercise objectives

WA5913.0

Notes:

Unit 10. Tuning and connection pool management problems

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 application server logs
- Enable tracing for connection manager components
- Interpret and analyze the trace data
- Explain how to use PMI to monitor connections

How you will check your progress

- Checkpoint questions
- Lab exercise

References

Support Authority: Real-time problem determination with WebSphere diagnostic providers:

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

Database Connection Pool Analyzer for IBM WebSphere Application Server:

<http://www.alphaworks.ibm.com/tech/jcp>

Unit objectives

After completing this unit, you should be able to:

- Identify connection pool problems
- Describe what to look for in the application server logs
- Enable tracing for connection manager components
- Interpret and analyze the trace data
- Explain how to use PMI to monitor connections

© Copyright IBM Corporation 2013

Figure 10-1. Unit objectives

WA5913.0

Notes:

Topics

- Connection pooling
- Troubleshooting configuration problems
- Troubleshooting connection leaks
- Troubleshooting stale connections

© Copyright IBM Corporation 2013

Figure 10-2. Topics

WA5913.0

Notes:

10.1.Connection pooling

Connection pooling



© Copyright IBM Corporation 2013

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

8.0

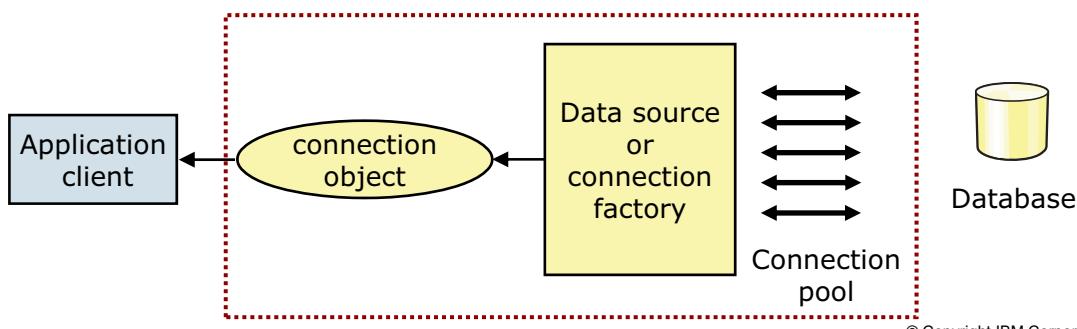
Figure 10-3. Connection pooling

WA5913.0

Notes:

What is connection pooling?

- The application server maintains a pool of ready-to-use connections to a data store
 - Application client requests a connection by 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



© Copyright IBM Corporation 2013

Figure 10-4. What is connection pooling?

WA5913.0

Notes:

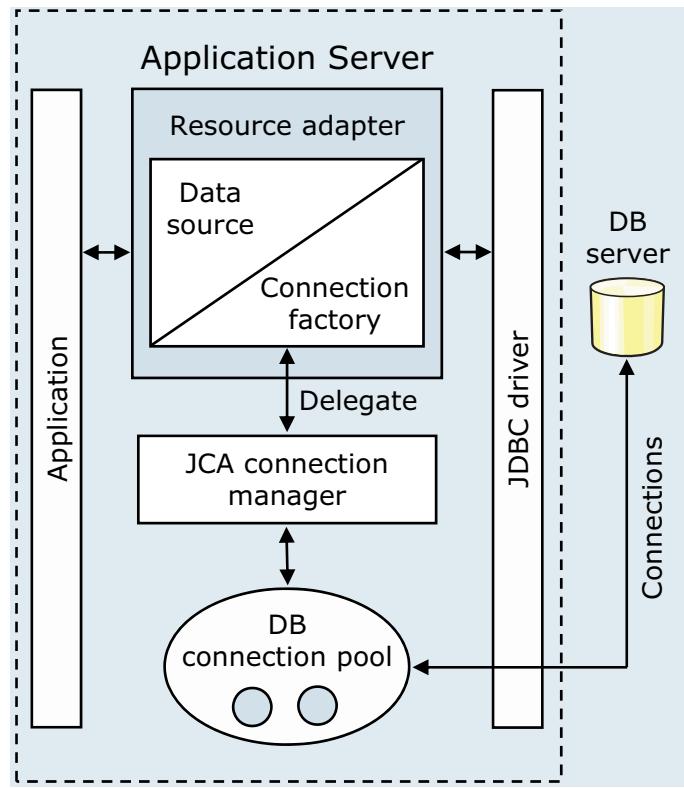
Creating a connection to a database is an expensive operation.

- 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.

JCA connection pooling architecture

- Connection pooling supports two components:
 - JCA connection manager
 - Relational resource adapter
- How it works:
 - An application acquires a data source (DS) or connection factory (CF) object from the resource adapter
 - The DS or CF delegates the connection allocation request to the connection manager
 - The 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



© Copyright IBM Corporation 2013

Figure 10-5. JCA connection pooling architecture

WA5913.0

Notes:

J2C is J2EE Connector architecture.

JDBC is Java Database Connectivity.

WebSphere Application Server implements the Java EE Connector Architecture (JCA) V1.5 specification.

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, the J2C connection pool manager implements these capabilities. 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, 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:

1. An application acquires a data source or connection factory object from the relational resource adapter.
2. The data source or connection factory delegates the connection allocation request to the JCA connection manager.
3. The JCA connection manager retrieves a free connection from the pool or creates a new one if none is available.
4. The retrieved or created connection is returned to the application.

Types of connection pools in WebSphere

- JDBC provider connection pool
 - Provides access to a relational database
 - Provides a connection by using a JDBC data source
 - Managed in the administrative console under JDBC providers
- JMS provider connection pool
 - Provides access to the data store used by the default messaging engine or a WebSphere MQ provider
 - Provides a connection by using a JMS connection factory
 - Managed in the administrative console under JMS providers
- EIS connection pool
 - Provides access to enterprise information systems such as CICS, earlier databases such as IMS, and other back-end systems
 - Provides a connection by using a data source or connection factory
 - Managed in the administrative console under Resource adapters
- The WebSphere **J2C connection pool manager** (JCA connection manager) manages all of these connection pools

© Copyright IBM Corporation 2013

Figure 10-6. Types of connection pools in WebSphere

WA5913.0

Notes:

JMS is Java Message Service.

EIS is enterprise information system.

CICS is Customer Information Control System.

IMS is Information Management System.

The J2C connection pool manager pools and manages all connections to a JCA-compliant data store. Support for WebSphere Version 4 data sources is also provided for compatibility with previous versions, but uses the old WebSphere connection manager architecture. This change is important because version 4 data sources have their own problem areas and generate their own set of error messages.

The WebSphere V8 *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.

Connection lifecycle

- A ManagedConnection object is always in one of three states: *DoesNotExist*, *InFreePool*, or *InUse*
- After a connection is created, it can be in either:
 - *InUse* (if allocated to an application)
 - *InFreePool* state
- Between the three states are transitions
- For example, you can make the transition from the *InFreePool* state to *InUse* state if:
 - Application has called the data source or connection factory `getConnection()` method
 - A free connection is available in the pool with matching properties
- After a connection is closed with the `close()` method, it is either:
 - Returned to the free pool
 - Destroyed

© Copyright IBM Corporation 2013

Figure 10-7. Connection lifecycle

WA5913.0

Notes:

Getting connections

The first set of transitions that are covered are the transitions in which the application requests a connection from either a data source or a connection factory. In some of these scenarios, a new connection to the database is created. In others, the connection might be retrieved from the connection pool or shared with another request for a connection.

DoesNotExist

Every connection begins its lifecycle in the *DoesNotExist* state. When an application server starts, the connection pool does not exist. Therefore, there are no connections. The first connection is not created until an application requests its first connection. More connections are created as needed, according to the guarding condition.

InFreePool

The transition from the *InFreePool* state to the *InUse* state is the most common transition when the application requests a connection from the pool.

InUse

The idea of connection sharing is seen in the transition on the InUse state. This transition indicates that if an application requests a shareable connection (`getConnection`) with the **same** sharing properties as a connection that is already in use (`ShareableConnectionAvailable`), the existing connection is shared.

Returning connections

All of the transitions explained previously involve getting a connection for application use. With that goal, the transitions result in a connection closing, and it either returns to the free pool or is destroyed. Applications should explicitly close connections (note: the connection that the user gets back is really a connection handle) by calling `close()` on the connection object.

For reference, see the connection lifecycle article in the information center:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/cdat_conlifcyc.html

Shareable versus unshareable connections (1 of 2)

- Shared connections (default)
 - Multiple `getConnection()` calls use a single connection within the 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
- Common issues
 - Shared connections might be held too long, exhausting the pool, showing symptoms of a connection leak

© Copyright IBM Corporation 2013

Figure 10-8. Shareable versus unshareable connections (1 of 2)

WA5913.0

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 receive a handle (indirectly) for the same physical connection to the resource. The benefits of this feature are improved performance and a reduction in the number of physical connections that must 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 that an application obtains within a local transaction containment are kept reserved. These connections are in the shared pool for use within that local transaction containment until it ends. This activity occurs even if the application explicitly closes the connection. This behavior is beneficial for many applications, but can have unintended consequences for others.

The use of shareable connections can provide performance benefits in many situations, but you must be aware of this default behavior and take it into account when developing applications. Otherwise, the application can 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 allowing only one active connection handle at a time.
- A connection can be held long after the application calls the `close()` method.

Shareable versus unshareable connections (2 of 2)

- Unshared connections:
 - Each `getConnection()` call results in a connection that is allocated from the connection pool
 - The connection is released back to the pool upon a `close()` call
 - Many different connections might be used within the same transaction
- Common issues:
 - Unshared connections might exhaust the pool as many might be used within a transaction
 - Wasted connections

© Copyright IBM Corporation 2013

Figure 10-9. Shareable versus unshareable connections (2 of 2)

WA5913.0

Notes:

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. When 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.

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 that is used to support J2EE 1.2 applications
J2CA or CWWJC	<ul style="list-style-type: none"> • Java EE connector (J2C Connection pool manager) • Most recent JCA 1.5 compliant connection manager
WSCL or CWWSC	<ul style="list-style-type: none"> • WebSphere client (Java EE 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

© Copyright IBM Corporation 2013

Figure 10-10. Detecting connection management-related problems

WA5913.0

Notes:

JCA connection management problems can appear in the WebSphere logs as error messages with signatures shown in the table shown.

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, V6, and V7 to provide support for Java Platform, Enterprise Edition 1.2 applications.

Typical connection pool runtime problem symptoms

- Sporadic failure to connect to an existing data source or connection factory:
 - The application was 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 whether 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

© Copyright IBM Corporation 2013

Figure 10-11. Typical connection pool runtime problem symptoms

WA5913.0

Notes:

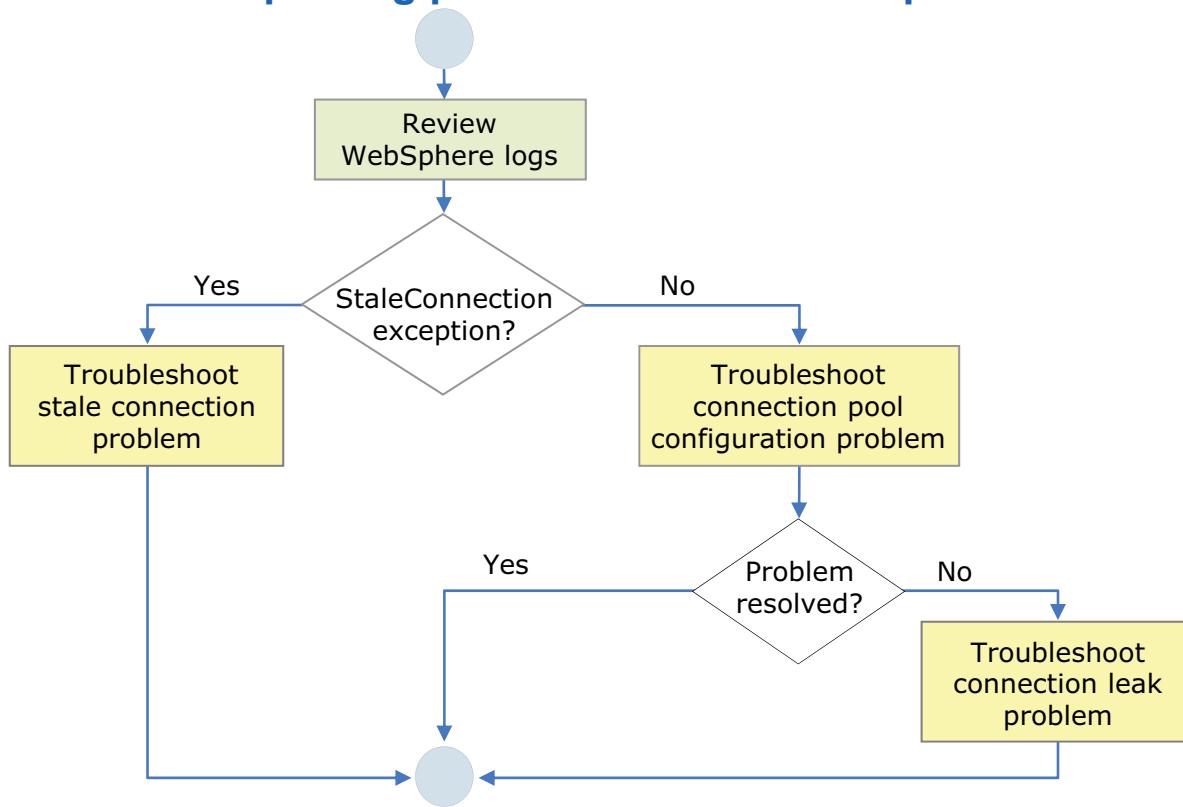
After configuration problems are resolved, you might begin to see sporadic failures during database access.

The main symptom that points to a possible connection pool problem is when the application is 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` files for any additional exception messages. The root cause is likely to be one of:

- Improperly tuned connection pool settings
- Connection leak
- Stale connection

Connection pooling problem determination path



© Copyright IBM Corporation 2013

Figure 10-12. Connection pooling problem determination path

WA5913.0

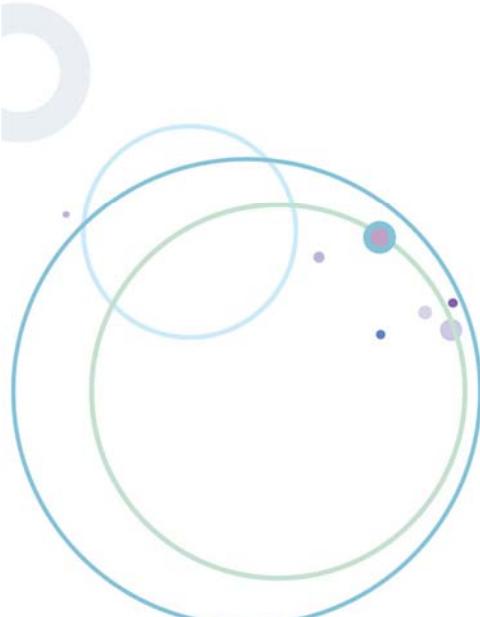
Notes:

This diagram shows the decision-based path to troubleshoot connection pooling problems. Keep in mind that before this point, you should confirm that you correctly configured connection parameters, and tested that you can properly establish a connection.

The next topics explain each troubleshooting task in detail.

10.2.Troubleshooting configuration problems

Troubleshooting configuration problems



© Copyright IBM Corporation 2013

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

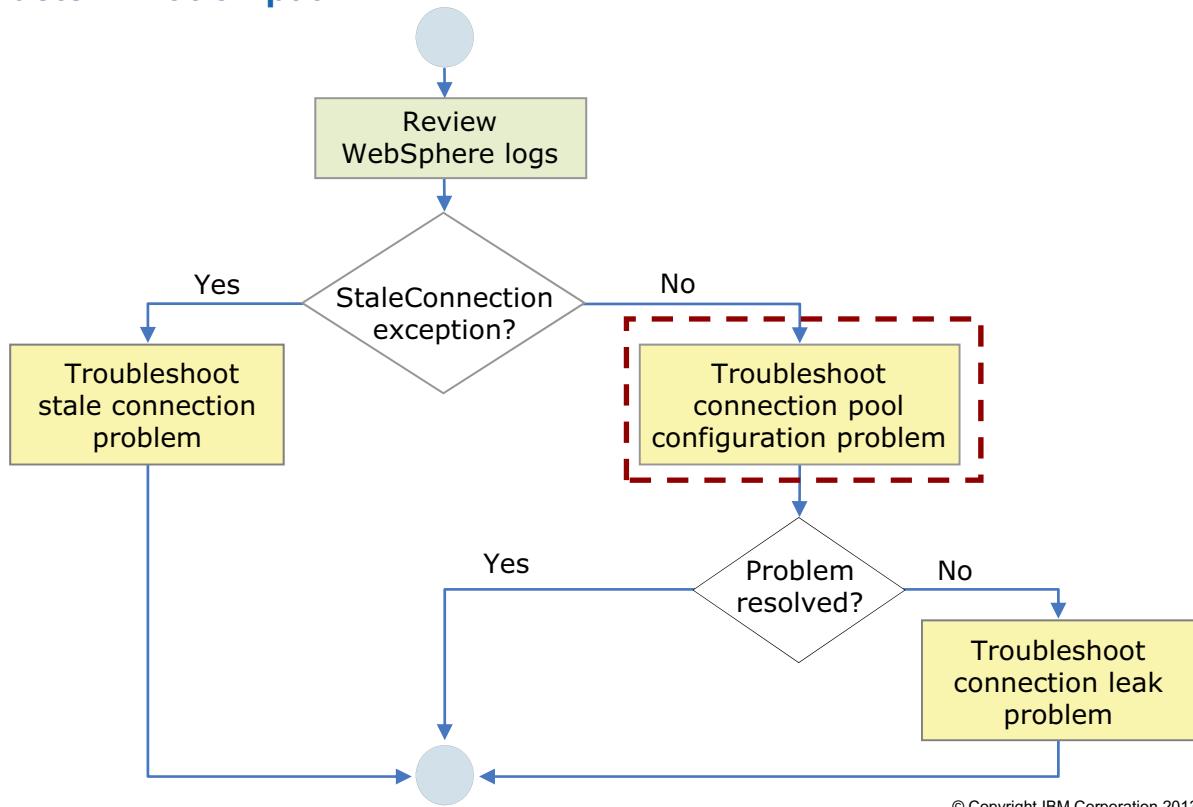
8.0

Figure 10-13. Troubleshooting configuration problems

WA5913.0

Notes:

Troubleshooting connection pool configuration in the problem determination path



© Copyright IBM Corporation 2013

Figure 10-14. Troubleshooting connection pool configuration in the problem determination path

WA5913.0

Notes:

When a `StaleConnectionException` does not accompany the symptom for a connection pool problem in the WebSphere logs, start your problem determination effort by looking at the connection pool configuration to rule out any performance tuning problems.

The need for tuning connection pools

- Keeping connections in a pool uses 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 must 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

© Copyright IBM Corporation 2013

Figure 10-15. The need for tuning connection pools

WA5913.0

Notes:

Keep in mind that 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 might start seeing problems too.

Sometimes running queries take longer than expected and causes `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 result.

An orphaned connection is one that is allocated to a client but is not being used and is therefore abandoned, most likely because the client is experiencing an unexpected problem.

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 (3 minutes)
 - A value of 0 instructs the pool manager to continue waiting until a connection becomes available

© Copyright IBM Corporation 2013

Figure 10-16. Key connection pool parameters

WA5913.0

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 the pool returns one that is “in use”. If the connection is not returned 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 is reached. If a connection is not available within this time, the pool manager throws a `ConnectionWaitTimeoutException`.



Connection pool parameters in the administrative console

- **Minimum connections**

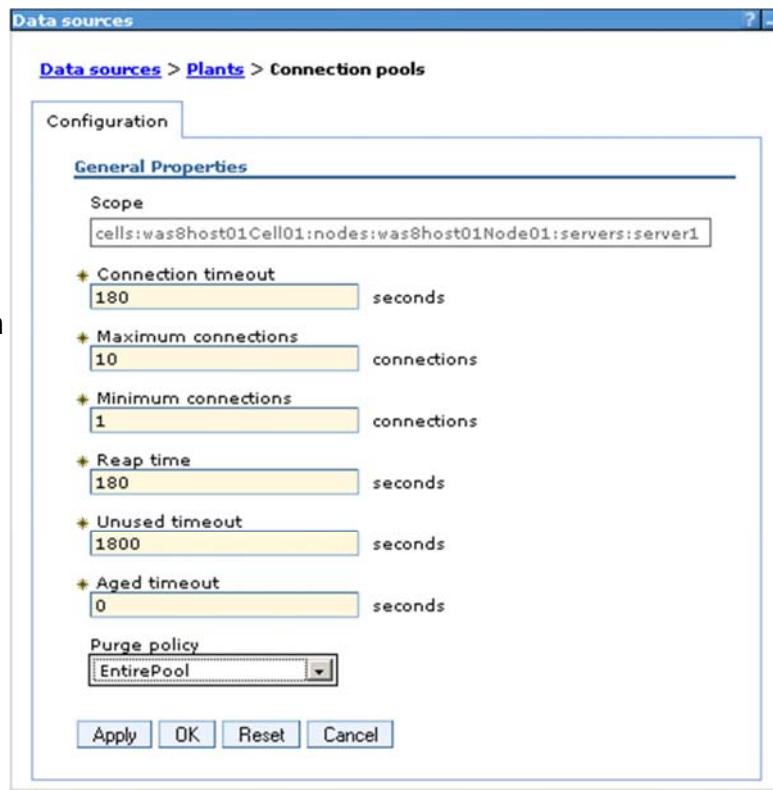
- Minimum number of physical connections to be maintained

- **Unused timeout**

- Interval (secs) after which an unused or idle connection is discarded from the pool

- **Purge policy** for stale connections

- *EntirePool* (the default)
- *FailingConnectionOnly*



© Copyright IBM Corporation 2013

Figure 10-17. Connection pool parameters in the administrative console

WA5913.0

Notes:

The following extra 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* are set to three, and one physical connection is created, the *unused timeout* thread does not discard that connection. By the same token, the thread does not automatically create two extra 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 are unused for longer than the time value specified in *unused timeout*. This process continues until the pool reaches

the number of connections that are specified in *minimum connections*. It also discards any connections 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 (2 minutes) is discarded. The *reap time* value affects the accuracy of this timeout, and its performance.

- **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. The *reap time* value affects the accuracy of this timeout, and its performance.

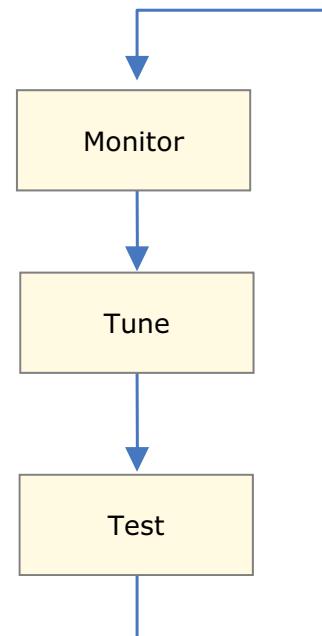
- **Purge policy:** Specifies how to purge connections when a stale connection or unrecoverable connection error is detected. Valid values are *EntirePool* (the default) and *FailingConnectionOnly*.

If you use *EntirePool*, all physical connections in the pool are destroyed when a stale connection is detected. Final destruction of connections that 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.

Task for tuning a connection pool

- Monitor connection pool runtime behavior
 - Enable PMI and select appropriate statistics set
 - View connection pool performance metrics by using Tivoli Performance Viewer (TPV)
 - Generate tuning advice by using TPV Performance Advisor
- Tune connection pool parameters
 - Make one change at a time
 - Apply recommendations and tuning advice
- Test application
 - Use a load generation tool to simulate production-like loads
 - Compare results with original baseline
 - Document results
- Repeat the monitor-tune-test cycle until the problem is resolved and performance goals are met



© Copyright IBM Corporation 2013

Figure 10-18. Task for tuning a connection pool

WA5913.0

Notes:

Performance tuning, in general, is an iterative and incremental process that consists 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. Some tools can assist with this practice, such as the *Tivoli Performance Advisor* embedded in WebSphere Application Server, 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 problem disappears and then backing them off until any wasted resources are recovered.

A runtime performance advisor, which is known as the Performance and Diagnostic Advisor, is also available.

The Performance Monitoring Infrastructure (PMI) is enabled by default.

Monitoring the connection pool by using Tivoli Performance Viewer

- These 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 might 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 used	<ul style="list-style-type: none"> Ensure that you are not consistently maxed at 100% Counter requires the selection of either <i>all</i> or <i>custom</i> PMI statistic set

© Copyright IBM Corporation 2013

Figure 10-19. Monitoring the connection pool by using Tivoli Performance Viewer

WA5913.0

Notes:

The PMI of WebSphere collects these metrics. These metrics are available for display in Tivoli Performance Viewer under the *JDBC Connection Pools* and *JCA Connection Pools* modules.

To access these modules, in the administrative console:

- Go to **Monitoring and Tuning > Performance Viewer > Current Activity**.
- Select the server that you want to monitor.
- Expand **Performance Modules > JDBC Connection Pools or JCA Connection Pools**.

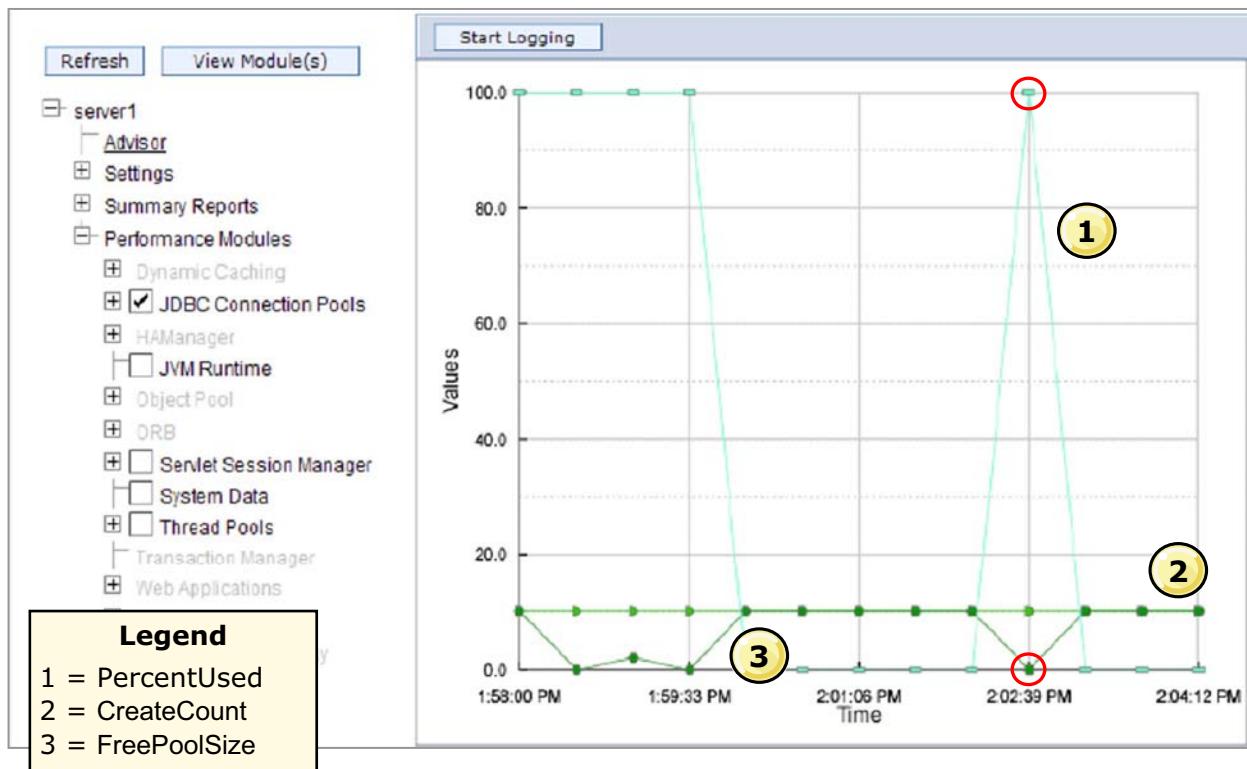
To view the collected metrics for a JDBC data source:

- 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

TPV connection pool monitoring example



© Copyright IBM Corporation 2013

Figure 10-20. Tivoli Performance Viewer connection pool monitoring example

WA5913.0

Notes:

This Tivoli Performance Viewer screen capture shows an example of connection pool graphs for a data source:

1. The cyan colored graph plots the *PercentUsed* metric (average percent of the pool that is in use).
2. The green graph plots the *CreateCount* metric (total number of connections created). Ten connections are created, reaching the default *maximum connections* value for the pool.
3. The dark green graph plots the *FreePoolSize* metric (number of free connections in the pool).

Notice that, as expected, when the *FreePoolSize* is 0, indicating no connection available in the pool, the *PercentUsed* value is at 100%.



Generating tuning advice by using TPV Performance Advisor

- TPV Performance Advisor can provide configuration advice for connection pool size
 - Tuning advice is provided in the Performance Advisor section of Tivoli Performance Viewer
 - Based on collected PMI data over the last 1-minute interval
 - Uses IBM-defined recommendations for advice basis
- Limitations:
 - Pool sizing advice might not be generated if your timeout values are too high (pools are not returning to minimum values)
 - Tivoli Performance Viewer Advisor gives recommendations only when processor usage is greater than or equal to 50%

© Copyright IBM Corporation 2013

Figure 10-21. Generating tuning advice by using TPV Performance Advisor

WA5913.0

Notes:

WebSphere provides two separate tuning advisors:

- TPV Performance Advisor: Runs on demand and outputs advice to the Tivoli Performance Viewer 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**.

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 is generated. This setting 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:

1. Go to **Monitoring and Tuning > Performance Viewer > Current Activity**.
2. Select the server for which you want to generate advice.
3. Select **Advisor** and look at the generated advice list.

Message

TUNE5033W: For data source jdbc/NoTxTradeDataSource, the maximum connection pool size is unusually large. Typically, the connection pool size is no more than 30.

Severity

Config

Description

In general, a very large connection pool reduces performance, although it might be necessary for some applications.

User Action

To change the connection pool properties, open the administrative console and click JDBC Providers > JDBC_provider > Data Sources > data_source > Connection pool properties. See the information center for more information about queueing.

Detail

Currently, the size of the minimum connection pool is 10 and the maximum pool size is 50.

© Copyright IBM Corporation 2013

Figure 10-22. TPV Performance Advisor example

WA5913.0

Notes:

The TPV Performance Advisor generates a table of messages according to different severity levels: Alert and Config. In this example the message is: TUNE5033W: For data source jdbc/NoTxTradeDataSource, the maximum connection pool size is unusually large. Typically, the connection pool size is no more than 30. The TPV Performance Advisor also provides Description, User Action, and Detail.

For this example:

- **Severity:** Config
- **Description:** In general, a large connection pool reduces performance, although it might be necessary for some applications.
- **User Action:** To change the connection pool properties, open the administrative console and click **JDBC Providers > JDBC_provider > Data Sources > data_source > Connection pool properties**. For more information about queuing, see the information center for the WebSphere Application Server.
- **Detail:** Currently, the size of the minimum connection pool is 10 and the maximum pool size is 50.

Tuning the connection pool

- Goal is to create a large enough pool to handle a peak load but does not take up too many system resources
 - Unused connections during non-peak periods can be controlled with the minimum connections parameter
- To tune the connection pool, you must 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 other SQL calls
- Key parameters to tune are maximum connections and connection timeout
- 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

© Copyright IBM Corporation 2013

Figure 10-23. Tuning the connection pool

WA5913.0

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 are discovered. In particular, the two parameters that have the greatest effect on correcting connection pool configuration errors are:

- Connection timeout
- Maximum connections

Suppose that 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). In this case, increasing only the number of available connections is not the best approach. Conversely, if the connections are short-lived, then increasing only their number can lead to an application server that is overloaded in other areas during a peak because the extra connections are unnecessarily using 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 occur during a peak load, an optimal configuration can be achieved. Thus you have optimal performance with the lowest possible administrative cost.

Suggestions for tuning connection pools

- Maximum connections setting:
 - As a first guess, try doubling the number of the *maximum connections*
 - If this change 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 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 the problem is resolved or the setting is at the highest value that the client can tolerate
- Before you increase the pool size, consult with the database administrator
 - Ensure that the database server is configured to handle the maximum pool size setting

© Copyright IBM Corporation 2013

Figure 10-24. Suggestions for tuning connection pools

WA5913.0

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 does not start with 10 connections. The value of 10 connections *minimum* is a low water mark. Until 10 connections are required concurrently, the pool contains only the maximum number of concurrent connections that are required up to that point. Therefore, if the number of concurrent connections never reaches six, then the pool contains six connections. The number of connections that are needed exceeds 10. The number of connections in the pool does not drop below 10 until the pool is cleaned out. In other words, after the reap time expires, all unused connections will be removed until the *minimum connections* threshold is reached.

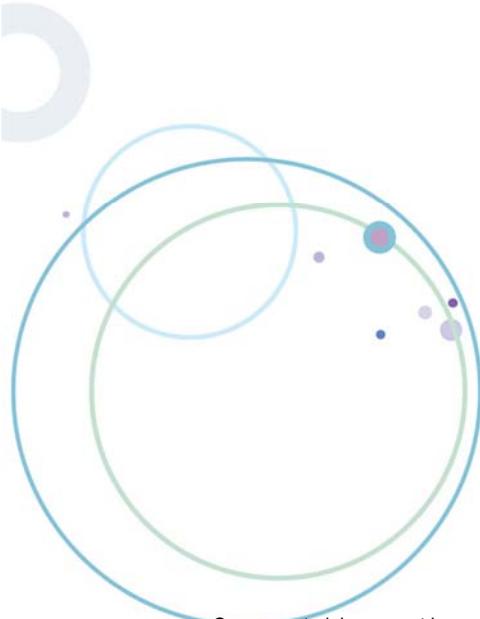
Configuration of the data sources should be done in consultation with the database administrator. For instance, the connection pool size should not be larger than the number of agents or connections that are allowed on the database server. This setting can become a problem, especially if cloning is used, because each application server allocates its own pool. To compute the maximum connections that the database can see, multiply the connection pool size by the size of the cluster. For example, assume that the connection pool maximum is 10 (the default), and you have a deployment of two instances of an application server on each of two 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 requires greater resources at the database server. If auto-indexing is activated, then the database might be spending much its time reindexing after each INSERT or UPDATE. If auto-indexing is turned off, then any SELECT operations might become more expensive because the indexes are stale. In either case, greater administrative cost is incurred for applications that are modifying the data in the database.

-
-

10.3.Troubleshooting connection leaks

Troubleshooting connection leaks



© Copyright IBM Corporation 2013

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

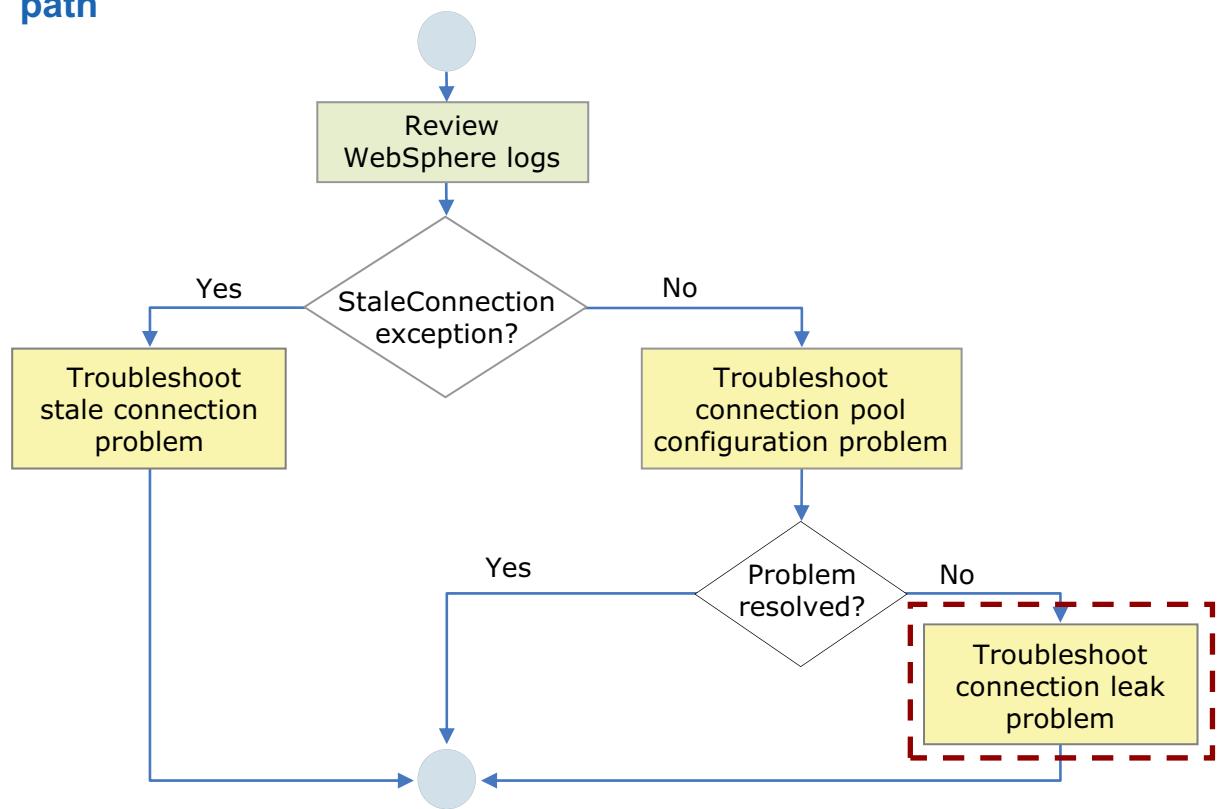
8.0

Figure 10-25. Troubleshooting connection leaks

WA5913.0

Notes:

Troubleshooting connection leaks in the problem determination path



© Copyright IBM Corporation 2013

Figure 10-26. Troubleshooting connection leaks in the problem determination path

WA5913.0

Notes:

After you rule out a stale connection and connection pool tuning problem, consider the possibility of a connection leak.

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

© Copyright IBM Corporation 2013

Figure 10-27. Connection leaks

WA5913.0

Notes:

A `ConnectionWaitTimeoutException` in the WebSphere logs is typically what identifies a connection leak. WebSphere eventually times out orphaned connections and returns them to the pool, but for an application that makes frequent use of database connections, it might not be enough. New connections can get queued up waiting for the database while old connections are waiting to be timed out. This delay can bring the application to a halt, and you can see `ConnectionWaitTimeoutExceptions`.

Common causes of connection leaks

- Poorly written applications often do not properly release database connections
 - Neglect to call `connection.close()`
 - Happens most often in an exception case
 - Connections should be closed in a `finally{}` block (try-catch-finally)
 - Also caused when one method gets a connection, calls multiple methods, and then forgets to close the connection when done
- Orphaned connections return to the pool only after timeout
 - Can cause a backup of new connections that are waiting for old connections to timeout
 - New connections that wait too long throw a `ConnectionWaitTimeoutException`

© Copyright IBM Corporation 2013

Figure 10-28. Common causes of connection leaks

WA5913.0

Notes:

Applications can suffer from performance problems and even appear to hang if they do not close their connections properly. This failure to close connections properly is most often because developers do not properly use the `connection.close()` method. To ensure that connections get closed properly, they should be closed in a `finally{}` block.

Connection leaks are traditionally 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.

Connection leak diagnosis tasks

- Configure the connection leak trace facility
 - Can be done by using the administrative console
- Run and monitor the application
 - Wait for ConnectionWaitTimeoutExceptions to occur
- Review and analyze trace file data
 - Locate the source of leak and resolve problem

© Copyright IBM Corporation 2013

Figure 10-29. Connection leak diagnosis tasks

WA5913.0

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 the application to user fewer connections

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
 - Helps you to narrow the search for the responsible source code
 - Lightweight with lower affect on performance than standard connection manager tracing (1 – 5% impact)
- Limitation:
 - Connection leak trace facility prints a stack trace of connections that are in use for more than 10 seconds

© Copyright IBM Corporation 2013

Figure 10-30. Connection leak trace facility

WA5913.0

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 throws a `ConnectionWaitTimeoutException`. When this exception is thrown, the connection leak tracer prints the stack traces for every open connection. It does so only when a problem occurs, providing instant recognition of when it occurred and reduced operating cost (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. It enables you to significantly narrow your search area when you look at the source code of the application to try to find the responsible code. It might also be helpful to IBM support because it helps 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. It prints the stack trace to the `trace.log` file. The default time interval is unchangeable. If you 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.

Configuring the connection leak trace facility

- Turned on by using a standard trace string `WAS.j2c=finest`
 - Includes `ConnLeakLogic` trace



© Copyright IBM Corporation 2013

Figure 10-31. Configuring the connection leak trace facility

WA5913.0

Notes:

As of WebSphere Application Server V6, the connection manager provides a diagnostic feature that is called the connection leak trace logic. It gathers information about which application methods might potentially lead to leaking connections (not closing connection) or holding onto connections longer than expected (for example, long-running queries).

To enable the connection leak trace logic, set the log detail level to: `ConnLeakLogic=finest`

Note: The `WAS.j2c` trace includes the `ConnLeakLogic` trace. If you enable the `WAS.j2c` trace, then you do not have to enable `ConnLeakLogic`.

Connection leak diagnostics are enabled in the administrative console by using the trace string that is shown. To do so:

1. Navigate to **Troubleshooting > Logs and Trace**.
2. Select the server where the application is running.
3. Select **Diagnostic Trace**.
4. Make sure that the **Enable Log** check box is checked.
5. Select **Change Log Detail Levels**.

6. Add the `j2c` trace string `WAS.j2c=finest` to the trace string text field.

What to look for in the trace

- Search `trace.log` for the string: Connection Leak Logic Information
- If present, some connections are in use for more than 10 seconds
- Analyze their stack trace to identify suspect application methods

```
Connection Leak Logic Information:
MCWrapper id 21171cb Managed connection WSRdbManagedConnectionImpl@1f69942 State:STATE_IDLE
Start time inuse Wed Mar 21 11:36:20 EDT 2012 Tim inuse 10 (seconds)
Last allocation time Wed Mar 21 11:36:20 EDT 2012
getConnection stack trace information:
    com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:1260)
    com.ibm.ws.rsadapter.jdbc.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:669)
    com.ibm.ws.webcontainer.datasource.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:636)
    com.ibm.connleak.LeakServlet doGet (LeakServlet.java:30)
    javax.servlet.http.HttpServlet.service (HttpServlet.java:575)
    javax.servlet.http.HttpServlet.service (HttpServlet.java:668)
    com.ibm.ws.webcontainer.servlet.ServletWrapper.service (ServletWrapper.java:1188)
    com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest (ServletWrapper.java:76)
    com.ibm.ws.webcontainer.servlet.ServletWrapper.handleRequest (ServletWrapper.java:45)
    com.ibm.ws.webcontainer.servlet.ServletWrapperImpl.handleRequest (ServletWrapperImpl.java:137)
    com.ibm.ws.webcontainer.filter.WebAppFilterManager.invokeFilters (WebAppFilterManager.java:947)
    com.ibm.ws.webcontainer.webapp.WebApp.handleRequest (WebApp.java:3703)
```

© Copyright IBM Corporation 2013

Figure 10-32. What to look for in the trace

WA5913.0

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.

Troubleshooting ConnectionWaitTimeoutException errors

- If an application is receiving the following messages:
 - com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException errors
 - J2CA0045E: Connection not available while invoking method queueRequest for resource jdbc/xxxxxx
- Then, it is possible that the database connections are not being returned to the connection pool in a timely manner
- In addition to tracing, it might be helpful to use wsadmin Jython commands to dump the contents of the connection pool at any point
 - The following example uses the PlantsByWebSphere data source

```
name = "Plants"
ds = AdminControl.queryNames("*:type=DataSource,name='"+name+"',*")
print AdminControl.invoke(ds, "showPoolContents")
```

© Copyright IBM Corporation 2013

Figure 10-33. Troubleshooting ConnectionWaitTimeoutException errors

WA5913.0

Notes:

It is also possible to put the commands into a script that can be invoked at a regular interval to check the status of the connection pool over time.

Output from “showPoolContents”

- The output shows
 - Free and running connections
 - Shared and unshared connections
- Run the command periodically to see whether any connections are still in use

```

PoolManager name:jdbc/PlantsByWebSphereDataSource
PoolManager object:-1297128224
Total number of connections: 10 (max/min 10/1, reap/unused/aged
180/1800/0, connectiontimeout/purge 30/EntirePool)

<. . .config info removed. . .>

Shared Connection information (shared partitions 200)

com.ibm.ws.LocalTransaction.LocalTranCoordImpl@89887ba5;RUNNING;
MCWrapper id 2e911c7a Managed connection
WSRdbManagedConnectionImpl@5de6780 State:STATE_TRAN_WRAPPER_INUSE
Used with transaction
com.ibm.ws.LocalTransaction.LocalTranCoordImpl@89887ba5;RUNNING;

com.ibm.ws.LocalTransaction.LocalTranCoordImpl@99e38198;RUNNING;
MCWrapper id e7f1b01b Managed connection
WSRdbManagedConnectionImpl@ab6de130 State:STATE_TRAN_WRAPPER_INUSE
Used with transaction
com.ibm.ws.LocalTransaction.LocalTranCoordImpl@99e38198;RUNNING;
. .
(0)(0)MCWrapper id 60016962 Managed connection
WSRdbManagedConnectionImpl@58ec3c1d State:STATE_ACTIVE_FREE
. .
(0)(0)MCWrapper id a6261a84 Managed connection
WSRdbManagedConnectionImpl@cceb5b8e State:STATE_ACTIVE_FREE
. .

UnShared Connection information
No unshared connections

```

© Copyright IBM Corporation 2013

Figure 10-34. Output from “showPoolContents”

WA5913.0

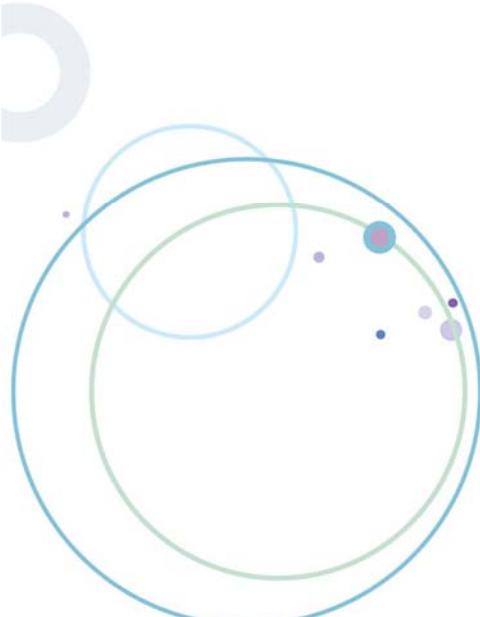
Notes:

For information on how to interpret the connection data, see the article, “How to troubleshoot J2CA0045E connection pooling problems”:

<http://www.ibm.com/support/docview.wss?uid=swg21385033>

10.4.Troubleshooting stale connections

Troubleshooting stale connections



© Copyright IBM Corporation 2013

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

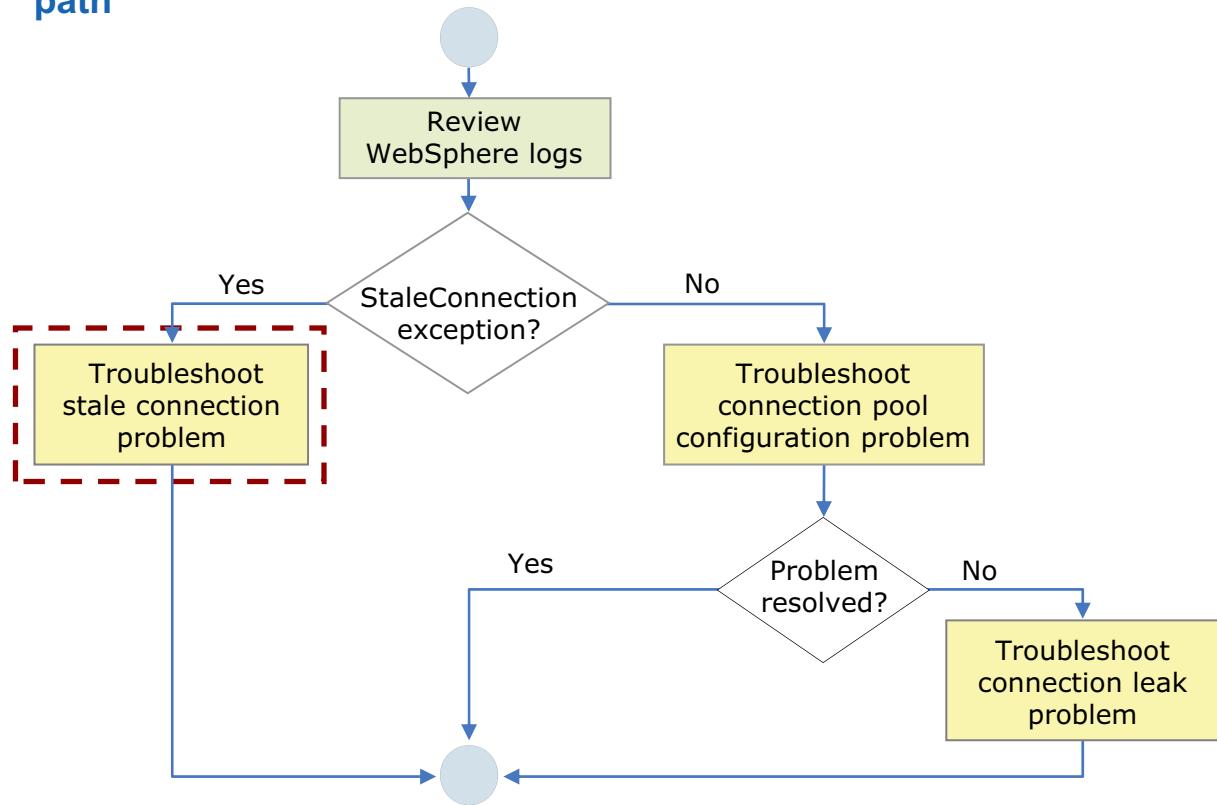
8.0

Figure 10-35. Troubleshooting stale connections

WA5913.0

Notes:

Troubleshooting stale connections in the problem determination path



© Copyright IBM Corporation 2013

Figure 10-36. Troubleshooting stale connections in the problem determination path

WA5913.0

Notes:

When you receive the exception `com.ibm.websphere.ce.cm.StaleConnectionException`, follow the resolution steps for troubleshooting a stale connection problem.

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:
 - A connection is no longer usable because of a database failure
 - An attempt is made to reuse an orphaned connection (applies only to WebSphere version 4.0 data sources)
 - WebSphere version 4.0 data source auto connection cleanup feature closed a connection and it is no longer usable

© Copyright IBM Corporation 2013

Figure 10-37. Stale connections

WA5913.0

Notes:

For a version 4.0 data source, a connection can be orphaned because it was not used 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. It 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 it, a *StaleConnection* exception occurs because the connection is already closed.

Recovering from a stale connection

- In general, a stale connection condition indicates that the connection to the database is no longer valid
 - The connection cannot be recovered and must be closed rather than be 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 purges the connection pool in accordance with its PurgePolicy setting and eliminates the bad connection
 - An administrator can purge a connection pool by using JCA lifecycle management
 - The application developer can explicitly catch a stale connection exception and programmatically recover from bad connections (for example, get a new one)

© Copyright IBM Corporation 2013

Figure 10-38. Recovering from a stale connection

WA5913.0

Notes:

Explicitly catching a *StaleConnection* exception while running within the context of a transaction does not cause the transaction to roll back. It provides the opportunity to programmatically recover from the exception, for example, by getting a new connection.



JCA lifecycle management

- Click **Resources > JDBC > Data sources**
- Select a data source and click **Manage state**

The screenshot shows the 'Data sources' interface under 'JCA lifecycle management'. At the top, there are buttons for 'Pause', 'Resume', and 'Purge'. Below them are icons for creating, deleting, and modifying resources. A search bar allows filtering by 'Name (JNDI name)' and 'Running object scope'. A status indicator shows 'Status' with a green circle. A table lists the resource 'Plants (jdbc/PlantsByWebSphereDataSource)' with details: Cell:washostCell01, Node:washostNode01, Server:server1, and a status icon. A total count of 1 is shown at the bottom.

Total	1
-------	---

- Click **Purge** to purge the contents of the connection pool for the data source or connection factory that is specified
 - Purging the pool does not affect ongoing transactions

© Copyright IBM Corporation 2013

Figure 10-39. JCA lifecycle management

WA5913.0

Notes:

Use this page to perform J2EE Connector Architecture (JCA) lifecycle operations on MBeans that correspond to the resources of your previous selection. These MBeans are compatible with version 6.0.2 or later of the application server. An MBean is listed for each running server that is in the scope of the configured resource. Pause or resume only those MBeans that are in the active state. Pausing an MBean halts outbound communication to the back end. This action also affects all applications that use the resource on the selected server.

The **Pause** button specifies to pause the MBean that is selected. Pausing an MBean halts outbound communication to the back-end resource, and this action affects all applications that use the resource on the selected server.

The **Resume** button specifies to resume the MBean that is selected. Resuming an MBean enables outbound communication to the back-end resource. This action affects all applications that use the resource on the selected server.

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 whether a specific query is getting the exception
 - Examine the SQLState and SQLCode returned with the exception
- Trace the problem by using one or more of the following options:
 - WAS.database
 - RRA
 - WAS.j2c

© Copyright IBM Corporation 2013

Figure 10-40. Other stale connection troubleshooting tasks

WA5913.0

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 re-create 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 the JDBC driver returns.



Connection pool troubleshooting tools (1 of 2)

- Diagnostic Provider ConnMgrDP for specific data sources
 - View state data
 - 16 resource values available for DB2 data sources

State Data Quick Link or Server Selection

[State Data Quick Link or Server Selection](#) > Diagnostic Providers > State Data

Preferences

State specification settings				
Node	Server	Name	Value	Description
was8host01Node01	server1	approximateNumberFreeConnections	0	null
was8host01Node01	server1	approximateNumberSharedConnections	10	null
was8host01Node01	server1	approximateNumberUnSharedConnections	0	null
was8host01Node01	server1	approximateNumberWaitingConnections	5	null

You can administer the following resources:

© Copyright IBM Corporation 2013

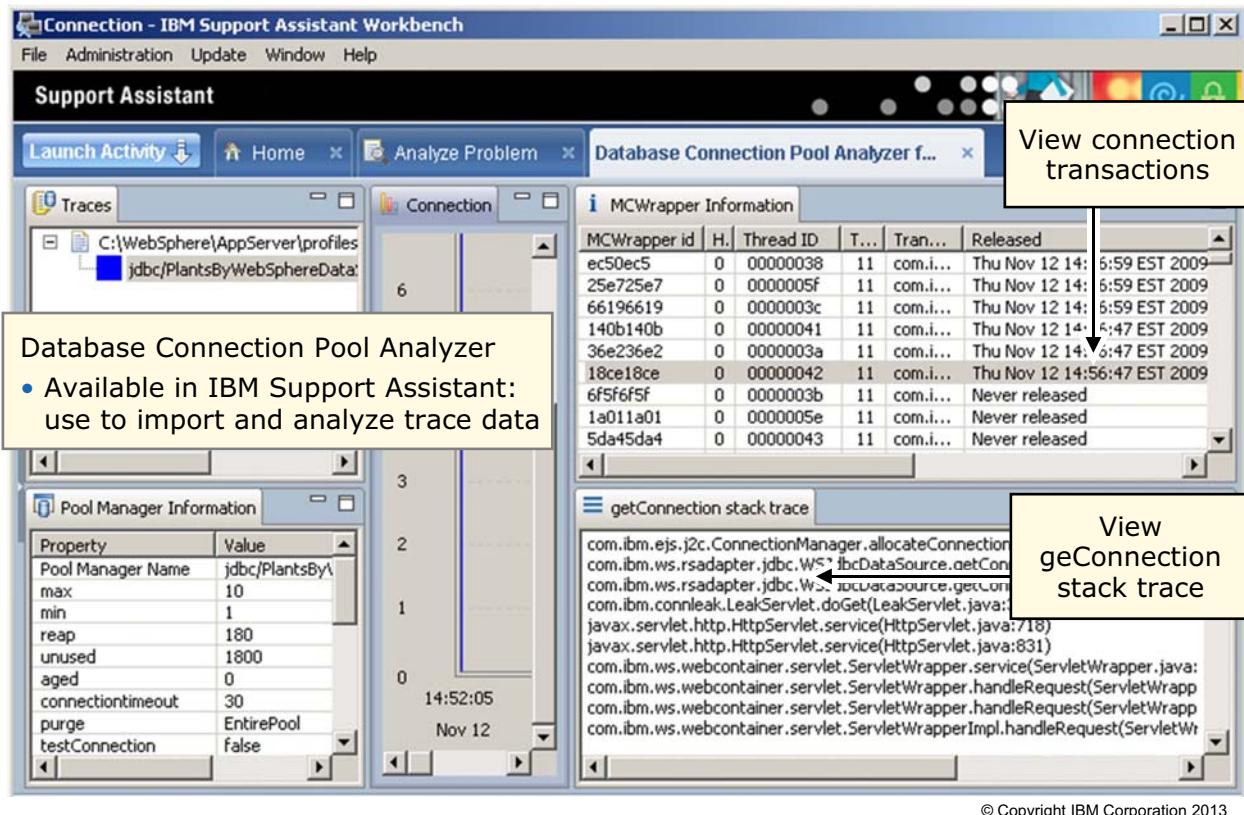
Figure 10-41. Connection pool troubleshooting tools (1 of 2)

WA5913.0

Notes:

The Diagnostic Provider utility is a simple front end in the administrative console that presents the available set of Diagnostic Providers and enables you to work with them. You can use it to view state data for specific data sources.

Connection pool troubleshooting tools (2 of 2)



© Copyright IBM Corporation 2013

Figure 10-42. Connection pool troubleshooting tools (2 of 2)

WA5913.0

Notes:

This tool employs a heuristic analysis engine to help you resolve problems that are related to Java Database Connectivity (JDBC) connection pools and find JDBC connection leaks.

Applications that are written in Java use JDBC to define how a client accesses database resources. JDBC connection pooling enables you to improve the response time of any application that requires connections, especially web-based applications. For example, when a user makes a request to a resource over the web, the resource accesses a data source. The total data storage that is required quickly rises when used in web-based applications, resulting in performance deterioration. However, when connection pooling capabilities are used, web applications are less affected. Performance improvements of up to 20 times can be made.

With connection pooling, most user requests do not incur the necessity of creating a new connection because the data source can locate and use an existing connection from the pool of connections. When the request is satisfied and the response is returned to the user, the resource returns the connection to the connection pool for reuse. The operation cost of a disconnection is avoided. Each user request incurs a fraction of the cost for connecting or disconnecting. After the initial resources are used to produce the connections in the pool, any other cost of operation is insignificant because the existing connections are reused.

For example, if a JDBC connection pool is deployed incorrectly and JDBC connections are not released after use, applications can suffer hangs, resource starvation, and other connection problems. Database Connection Pool Analyzer for IBM WebSphere Application Server helps you analyze JDBC connection pool leaks and resolve JDBC connection pool-related problems, particularly when using WebSphere Application Server.

How does it work?

The tool analyzes JDBC connection pool manager traces and provides the following functions:

- Analysis of JDBC data source
- Analysis of JDBC connection pool configuration
- JDBC connection chart
- View Java stack trace
- View of the `getConnection` method

The tool performs the following functions:

- Parses trace files for JDBC connection pool manager
- Detects available JDBC data sources and configurations
- Counts the number of waiters for connections and the number of connections
- Finds Java stack traces associated with JDBC connection leaks
- Provides charts of JDBC connections for each JDBC data source

Unit summary

Having completed this unit, you should be able to:

- Identify connection pool problems
- Describe what to look for in the application server logs
- Enable tracing for connection manager components
- Interpret and analyze the trace data
- Explain how to use PMI to monitor connections

© Copyright IBM Corporation 2013

Figure 10-43. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. True or false: Two key connection pool parameters for tuning are maximum connections and connection timeout.
2. True or false: The purge policy specifies how to purge connections when a stale connection or unrecoverable connection error is detected.
3. True or false: A connection leak arises when the maximum connections parameter is set to zero.

© Copyright IBM Corporation 2013

Figure 10-44. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. True
2. True
3. False: A connection leak is a situation that arises when the application that uses allocated connections does not return them back to the pool after use.

© Copyright IBM Corporation 2013

Figure 10-45. Checkpoint answers

WA5913.0

Notes:

Exercise 8



Troubleshooting a connection leak

© Copyright IBM Corporation 2013

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

8.0

Figure 10-46. Exercise 8

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Examine tuning parameters for data source connection pools
- Use a diagnostic provider to examine state data for a data source
- Recognize a tuning parameter problem situation
- Configure tracing and examine the logs

© Copyright IBM Corporation 2013

Figure 10-47. Exercise objectives

WA5913.0

Notes:

Unit 11. WebSphere security configuration problems

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 tracing on relevant security components
- Analyze and interpret trace information
- Recognize symptoms of common Secure Sockets Layer (SSL) configuration problems
- Recognize symptoms of common Java 2 security problems
- Locate the security configuration files
- Use tools to validate the security configuration files
- Use wsadmin securityoff to disable security

How you will check your progress

- Checkpoint questions
- Lab exercise

References

IBM WebSphere Application Server V8 Concepts, Planning, and Design Guide, SG24-7957-00, Chapter 12. Security

WebSphere Application Server V8 Information Center:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp>

Section: “Troubleshooting security”

IBM Redbook: WebSphere Application Server V7.0 Security, SG24-7660-00

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 tracing on relevant security components
- Analyze and interpret trace information
- Recognize symptoms of common Secure Sockets Layer (SSL) configuration problems
- Recognize symptoms of common Java 2 security problems
- Locate the security configuration files
- Use tools to validate the security configuration files
- Use wsadmin securityoff to disable security

© Copyright IBM Corporation 2013

Figure 11-1. Unit objectives

WA5913.0

Notes:



Topics

- Review of security components and security flows
- Common problems and troubleshooting methods
- SSL problems
- Java 2 security problems
- More tools and techniques

© Copyright IBM Corporation 2013

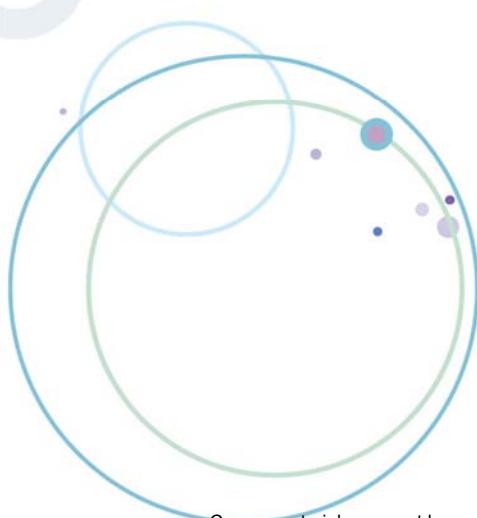
Figure 11-2. Topics

WA5913.0

Notes:

11.1. Review of security components and security flows

Review of security components and security flows



© Copyright IBM Corporation 2013

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

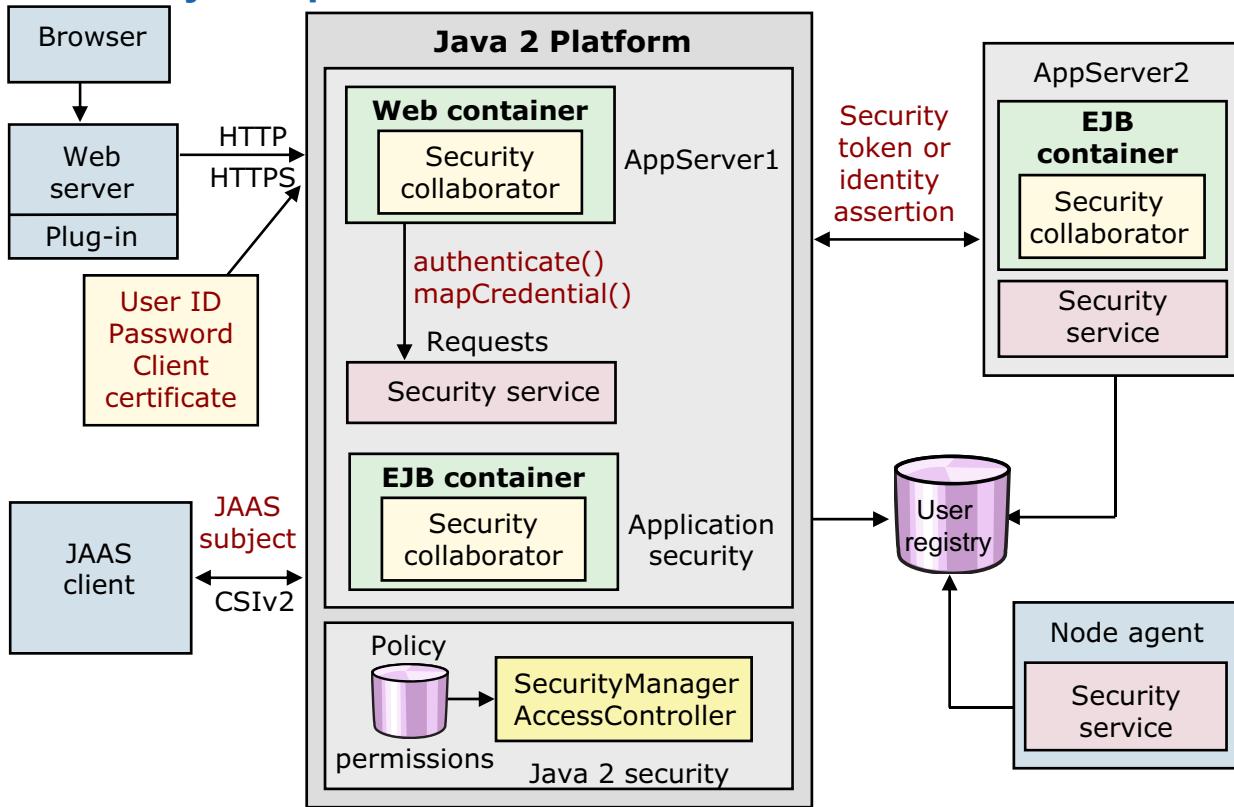
8.0

Figure 11-3. Review of security components and security flows

WA5913.0

Notes:

Security components overview



© Copyright IBM Corporation 2013

Figure 11-4. Security components overview

WA5913.0

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 run 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.

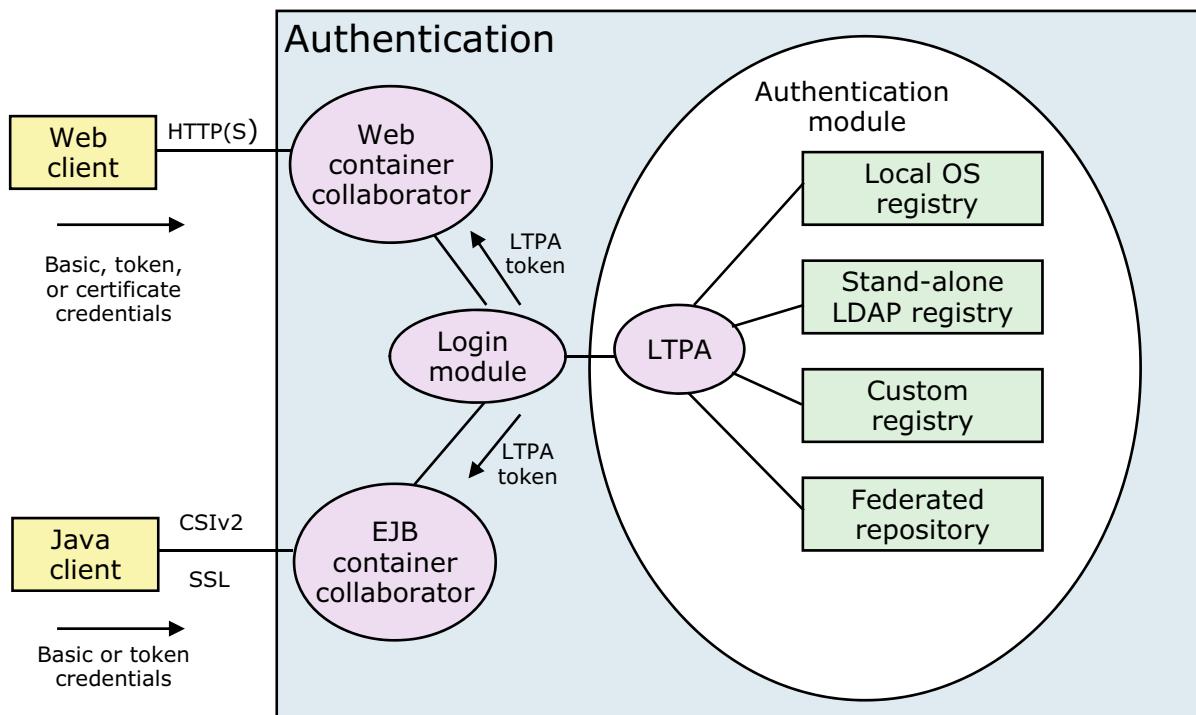
Security collaborators

A security collaborator is an application server component that enforces security constraints that the deployment descriptors specify. 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** runs in the web container and provides the following services to the application:
 - Checks authentication

- Checks authorization according to the constraint specified in the deployment descriptor
- Logs security tracing information
- The **EJB security collaborator** runs 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 do the following functions:
 - Checks authorizations according to the specified security constraint
 - Supports communication with local user registry
 - Logs security tracing information

Authentication flows



© Copyright IBM Corporation 2013

Figure 11-5. Authentication flows

WA5913.0

Notes:

The Simple WebSphere Authentication Mechanism (SWAM) is deprecated as of WebSphere Application Server V6.1.

The EJB security collaborator uses Common Secure Interoperability Version 2 (CSIV2) and Secure Authentication Service (SAS) to authenticate Java client requests to enterprise beans.

CSIV2 security: CSIV2 is an IIOP-based, three-tiered, security protocol that the Object Management Group (OMG) developed. This protocol provides message protection, interoperable authentication, and delegation in the following layers:

- A base transport security layer
- A supplemental client authentication layer
- A security attribute layer

Any calls that are made among secure Object Request Brokers (ORBs) are called over the CSIV2 security protocol, which sets up the security context and the necessary quality of protection.

After the session is established, the call is passed up to the enterprise bean layer.

**Important**

Secure Authentication Service (SAS) is supported only between WebSphere Application Server V6 and previous version servers that are federated in a V6.1 cell.

LTPA: LTPA is intended for single and multiple application server and host environments as the default user authentication protocol. It supports forwardable credentials and SSO. LTPA can support security in a composite environment through cryptography. The LTPA token contains authentication-related data, which is encrypted, digitally signed, and securely transmitted.

Later, at the receiving side, the information is decrypted, and the signature is verified. When using LTPA, a token is created with the user information and an expiration time. The keys sign this token. The LTPA token is time sensitive. All product servers that participate in a protection domain must have their time, date, and time zone synchronized. If they are not synchronized, LTPA tokens appear to be prematurely expired and cause authentication or validation failures. When SSO is enabled, this token is passed to other servers through cookies for web resources.

If the server and the client share keys, the token can be decrypted to obtain the user information. The WebSphere Application Server validates the data to ensure that the data is not expired and that the user information in the token is valid. On successful validation, the resources in the receiving servers are accessible after the authorization check. All WebSphere Application Server processes in a cell (deployment manager, node agents, or application servers) share a set of keys.

If key sharing is required between different cells, export them from one cell and import them to the other. For security purposes, a password is necessary to access the keys.

Security flows: Web browser communication

When a web browser sends a request to a WebSphere application, the following security interactions occur:

1. A user requests a web resource that is protected
2. The web server plug-in receives the request and recognizes that the resource is on the application server
3. Plug-in redirects the request to the web container security collaborator, which calls the JAAS login configuration if necessary
4. If authentication is successful, request reaches the web container
5. The web security collaborator passes the following to the security server for authorization:
 - User credentials
 - Security information that is contained in the deployment descriptor from EAR file
6. If the web application later calls an EJB:
 - User credentials are extracted from the established security context
 - The EJB collaborator does authorization

© Copyright IBM Corporation 2013

Figure 11-6. Security flows: web browser communication

WA5913.0

Notes:

Regarding the third point above, the web authenticator does not necessarily do 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 calls the JAAS login configuration and runs all the login modules in that configuration. The result of the login modules determines whether the user is authenticated or not.

Security flows: Java client communication

When a Java client interacts with a WebSphere application, the following occurs:

- 1.A Java client does a JAAS login before a business request
- 2.The CSIV2 or IBM SAS interceptor does authentication on the server-side on behalf of the ORB, and sets the security context
- 3.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

© Copyright IBM Corporation 2013

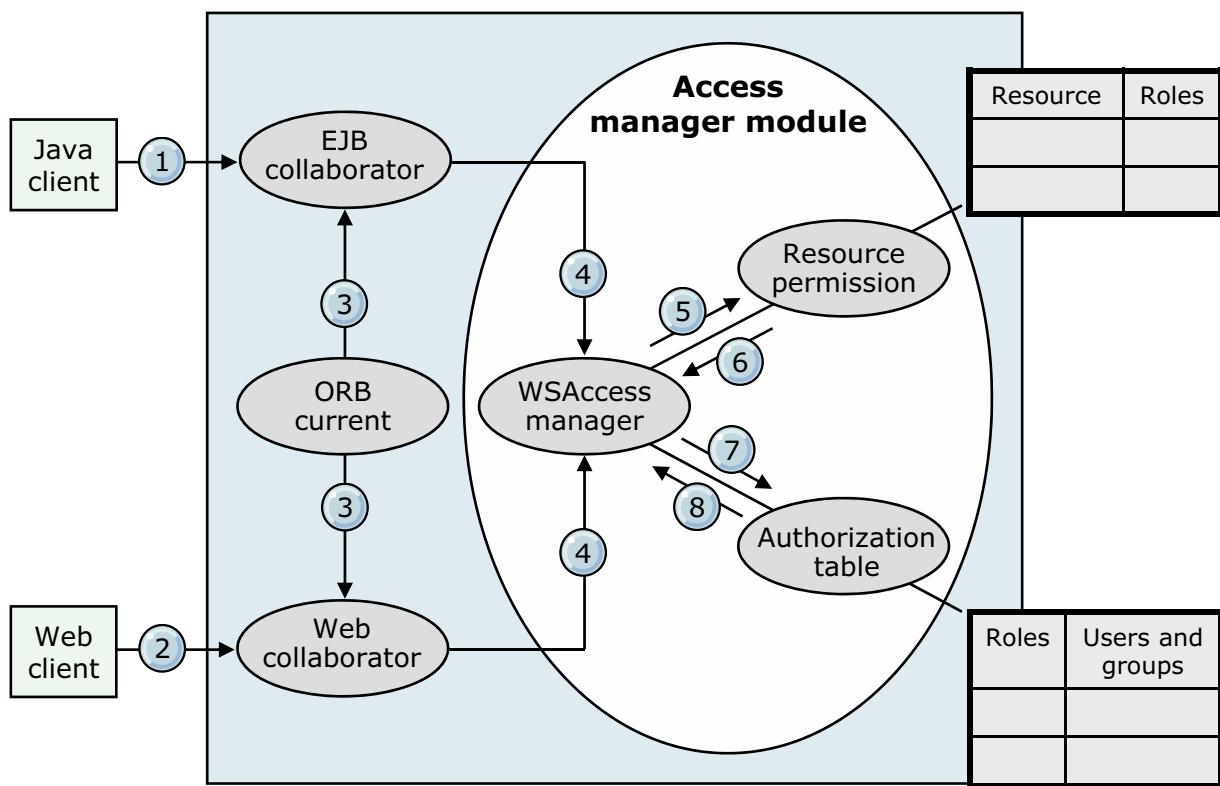
Figure 11-7. Security flows: Java client communication

WA5913.0

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; then the server initiates the authentication process. EJB clients explicitly run a JAAS login independent of (and before) making a business request. Authentication and authorization for an EJB client are done on separate calls.

Authorization flows



© Copyright IBM Corporation 2013

Figure 11-8. Authorization flows

WA5913.0

Notes:

1. EJB resource access – CSIV2 SAS, TCP/IP, SSL
2. Web resource access – HTTP or HTTPS
3. Received credentials
4. Resources and credentials
5. Resource
6. Roles
7. Roles and credentials
8. true or false

Federated repositories

- Provides federation capabilities by 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
 - Local operating system (as of V8.5)

© Copyright IBM Corporation 2013

Figure 11-9. Federated repositories

WA5913.0

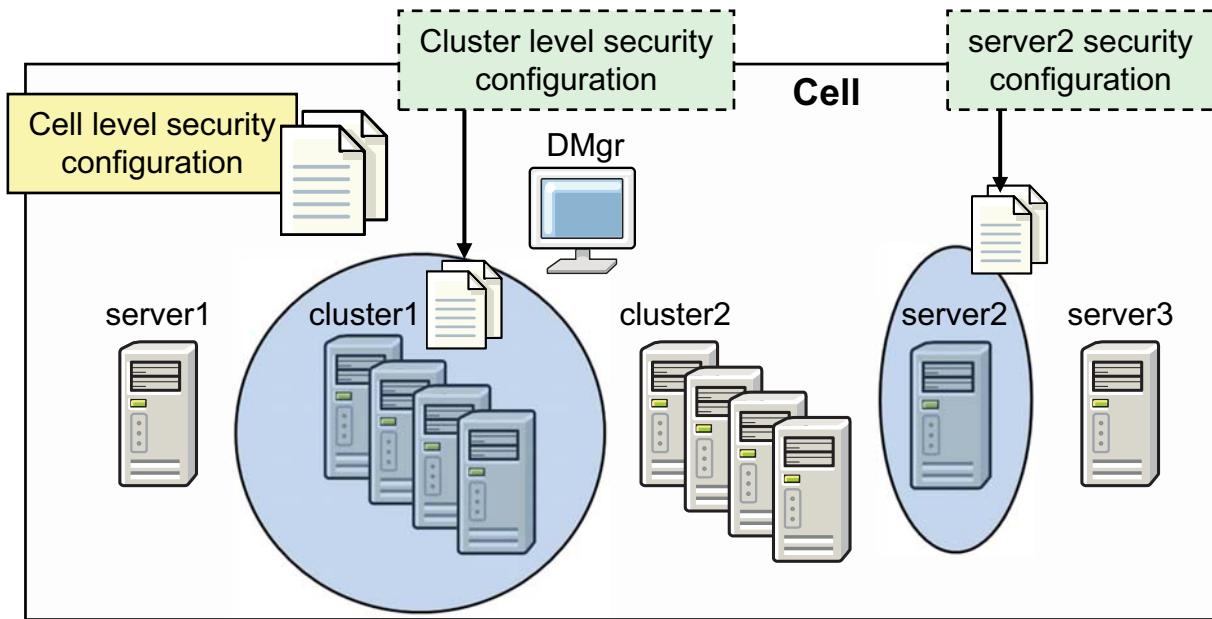
Notes:

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 with the federated repositories option.

When WebSphere Application Server is installed, the VMM component is automatically installed.

Security domains

- With security domains, it is possible to have not only a cell level security configuration, but also have multiple other security configurations that are scoped at different levels



© Copyright IBM Corporation 2013

Figure 11-10. Security domains

WA5913.0

Notes:

Security domains allow security to be defined at multiple levels, not just at the cell level. With security domains, it is possible to define one set of security settings for one application server and another set of configurations for a second application server. The only user that is assigned to the administrator role can configure multiple security domains. For example: with security domains, it is possible to have different user registries that are configured for distinct parts of the cell.

11.2.Common problems and troubleshooting methods

Common problems and troubleshooting methods



© Copyright IBM Corporation 2013

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

8.0

Figure 11-11. Common problems and troubleshooting methods

WA5913.0

Notes:

What can go wrong? The short list (1 of 2)

- Errors trying to enable administrative security
 - Invalid user IDs
 - Problems accessing the user registry
- Errors after security is enabled
 - Authentication failures
 - Authorization errors when accessing a web page
- Access and login problems after security is enabled
 - Problems trying to log on to the administrative console
 - Access exceptions if applications are not prepared for Java 2 security
 - Remote user registry inaccessible
 - Node synchronization problems

© Copyright IBM Corporation 2013

Figure 11-12. What can go wrong? The short list (1 of 2)

WA5913.0

Notes:

Many security-related problems like the ones that are listed on this slide are documented in detail in the WebSphere V8.0 Information Center in the “Troubleshooting security configurations” section.

What can go wrong? The short list (2 of 2)

- 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 and group role mappings
- Server fails to start

© Copyright IBM Corporation 2013

Figure 11-13. What can go wrong? The short list (2 of 2)

WA5913.0

Notes:

Many security-related problems like the ones that are listed on this slide are documented in detail in the WebSphere V8.0 Information Center in the “Troubleshooting security configurations” section.

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 occurs only when security is enabled does not always make it a security problem
 - More troubleshooting is necessary to ensure that the problem is security-related
 - Does the problem go away when Java 2 security is disabled?
- Did security seem to initialize properly?
 - Much security code runs during server initialization
 - Examine the `SystemOut.log` and `SystemErr.log` files to check for warnings and exceptions that are security-related

© Copyright IBM Corporation 2013

Figure 11-14. Approach to troubleshooting security-related issues

WA5913.0

Notes:

Here are some questions to ask when approaching security-related problems. Does the problem occur when security is disabled? This test is good to determine whether a problem is security-related or not. Just because a problem occurs only when security is enabled does not always make it a security problem; more troubleshooting is necessary to ensure that the problem is related to security. Does the problem go away when Java 2 security is disabled? Did security seem to initialize properly? Much security code runs during server initialization. Examine the `SystemOut` and `SystemErr` logs to check for warnings and exceptions that are related to security.

Normal security initialization messages

- Messages that are 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
  
```

© Copyright IBM Corporation 2013

Figure 11-15. Normal security initialization messages

WA5913.0

Notes:

The sequence of messages that are generated in the `SystemOut.log` indicates normal code initialization of an application server.

Note: In an ND cell, the `SystemOut.log` for all the servers (deployment manager, node agents, application servers) should have security-related entries identical to entries that are shown here.

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 are using a stand-alone LDAP registry, the message would be something like: `JSAS0004I: Principal name:<ldapserver>:389/`

Before V6.1, the actual server principal name was shown in this message.

Authentication or authorization problem

Many security problems fall under one of two categories

- **Authentication** is the process of determining who the caller is
 - When authentication fails, the reason 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 permission to invoke the requested method
 - When authorization fails, the cause is usually related to:
 - Application bindings from assembly and deployment
 - Identity of the caller who is accessing the method
 - Roles that are required to call the method

© Copyright IBM Corporation 2013

Figure 11-16. Authentication or authorization problem

WA5913.0

Notes:

Many security problems fall under one of two categories: authentications or authorization. Authentication is the process of determining who the caller is. When authentication fails, it is typically because the authentication data is not what was expected (wrong ID, wrong password). Either the user who is being authenticated is not in the registry, or the password is no longer valid, or the registry is misconfigured or not accessible.

Authorization is the process of validating that the caller has permission to run the requested method. When authorization fails, it is usually related to application bindings from assembly and deployment, the identity of the caller who is accessing the method, or the roles that the method requires.

Problems that are related to Secure Sockets Layer (SSL)

- SSL is a distinct layer of security
 - Problems are usually separate from authentication and authorization
- Certificate expiration issues
 - WebSphere monitors and automatically handles expiring certificates
- 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

© Copyright IBM Corporation 2013

Figure 11-17. Problems that are related to Secure Sockets Layer (SSL)

WA5913.0

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 started from your application server or deployment manager's `bin` directory.

As of WebSphere Application Server V6.1, most of the iKeyman function is incorporated into the administrative console. The iKeyman tool is still available as a stand-alone application.

See the information center topic, *Configuring certificate expiration monitoring*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.base.doc%2Finfo%2Faes%2Fae%2Ftsec_sslconfcert_expmon.html

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 might pinpoint the root cause
- Other times, it gives only a clue, and might be misleading
- When product support analyzes a stack trace, and it is not clear what the problem is:
 - They might request that you gather more trace data
 - You might also trace several security components with different levels of detail

© Copyright IBM Corporation 2013

Figure 11-18. Stack trace in the system log file

WA5913.0

Notes:

A single stack trace tells a lot about the problem: what code initiated the code that failed, what component is failing, or which class the failure came from. Sometimes the stack trace is all that is needed to solve the problem. It can pinpoint the root cause. Other times, it gives only a clue, and can be misleading. When IBM support analyzes a stack trace and it is not clear what the problem is, they might request that you gather more trace data. You might need to trace several security components with different levels of detail.

Example: SystemOut.log stack trace

- **Symptom:** The deployment manager seems to start successfully. However, an examination of the `SystemOut.log` file of the deployment manager shows many exceptions and stack traces.
- The beginning of the first stack trace looks like:

```
LdapRegistryI E SECJ0352E: Could not get the users  
matching the pattern wsbind because of the following  
exception javax.naming.CommunicationException: ldaphost: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)
```

© Copyright IBM Corporation 2013

Figure 11-19. Example: SystemOut.log stack trace

WA5913.0

Notes:

Notice that the LdapRegistry component logged this exception. The message provides important information for troubleshooting: the name of a user, wsbind, and the name and port of the LDAP server, ldaphost:389.

Example: SystemOut.log exceptions

- An attempt to log on to 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: ldaphost:389 [Root exception is java.net.ConnectException: Connection refused: connect].
```

```
FormLoginExte E SECJ0118E: Authentication error during authentication for user wasadmin
```

© Copyright IBM Corporation 2013

Figure 11-20. Example: SystemOut.log exceptions

WA5913.0

Notes:

If the exception in the deployment manager log is ignored and a user tries to log in to the administrative console, the following messages appear in the `SystemOut.log`. Again the messages contain the user ID for which the authentication failed and the address of the LDAP server. Since it is a different user ID this time, you might want to verify access to the LDAP server and whether LDAP is running on the remote host.

In all of these messages, what matters most is the “Root exception”, which is the cause of whatever bad things follow. In this case, apparently there is not a valid connection to the LDAP server.

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.ejb.*
 - com.ibm.ws.security.policy.*
 - com.ibm.ws.security.registry.*
 - others
- **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.*

© Copyright IBM Corporation 2013

Figure 11-21. Tracing security components

WA5913.0

Notes:

This slide shows many of the security components that can be traced. If you need to gather more data for a security-related problem, it is a good idea to try to pinpoint specific components. Trace components such as registry, web, or EJB are based on other symptoms that you might be seeing. It is a good idea to consult IBM Support's MustGather documents to learn which components to trace for specific problems.

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 EJB 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 to 10 in the Diagnostic Trace Service

© Copyright IBM Corporation 2013

Figure 11-22. MustGather tracing requirements

WA5913.0

Notes:

The MustGather documentation provides trace requirements for the following 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.

Result from security components trace

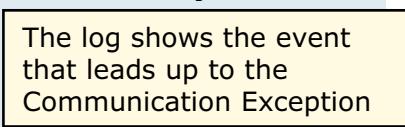
Portion of trace.log file

```

LdapRegistryI > getUniqueUserId 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://ldaphost:389
LdapRegistryI 3 enterJNDI:P=231764:O=0:CT
LdapRegistryI 3 exitJNDI:P=231764:O=0:CT
LdapRegistryI 3 javax.naming.CommunicationException: ladaphost:389
[Root exception is java.net.ConnectException: Connection refused:
connect]
LdapRegistryI A SECJ0418I: Cannot connect to the LDAP server
ldap://ldaphost:389.
    at java.net.PlainSocketImpl.socketConnect(Native Method)

```

The log shows the event
that leads up to the
Communication Exception



© Copyright IBM Corporation 2013

Figure 11-23. Result from security components trace

WA5913.0

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.

11.3. SSL problems

SSL problems



© Copyright IBM Corporation 2013

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

8.0

Figure 11-24. SSL problems

WA5913.0

Notes:

SSL use in WebSphere

- WebSphere uses SSL to provide data encryption and authentication between a client and server
 - Includes connections to resources outside of WebSphere like LDAP and databases
- WebSphere uses Java Secure Sockets Extension (JSSE) as the SSL implementation that the IBM JRE provides
 - JSSE handles the SSL handshake and protection that SSL provides
- SSL can be configured between many different endpoints in WebSphere
 - Browser to web server
 - Web server plug-in to the application servers
 - Application servers to LDAP servers
 - Application servers to databases

© Copyright IBM Corporation 2013

Figure 11-25. SSL use in WebSphere

WA5913.0

Notes:

The Java Secure Sockets 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 functions for data encryption, server authentication, message integrity, and optional client authentication.

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

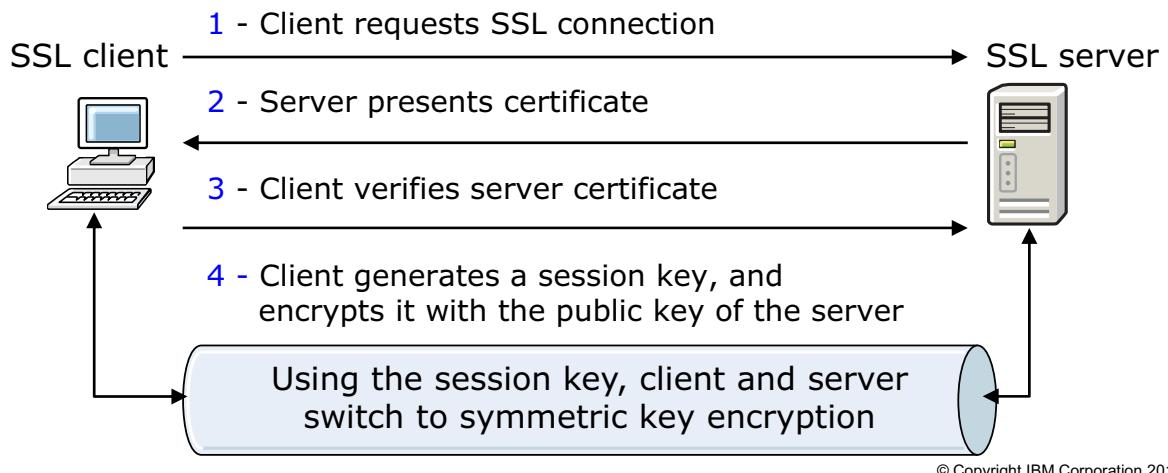


Figure 11-26. How does SSL work? The “handshake”

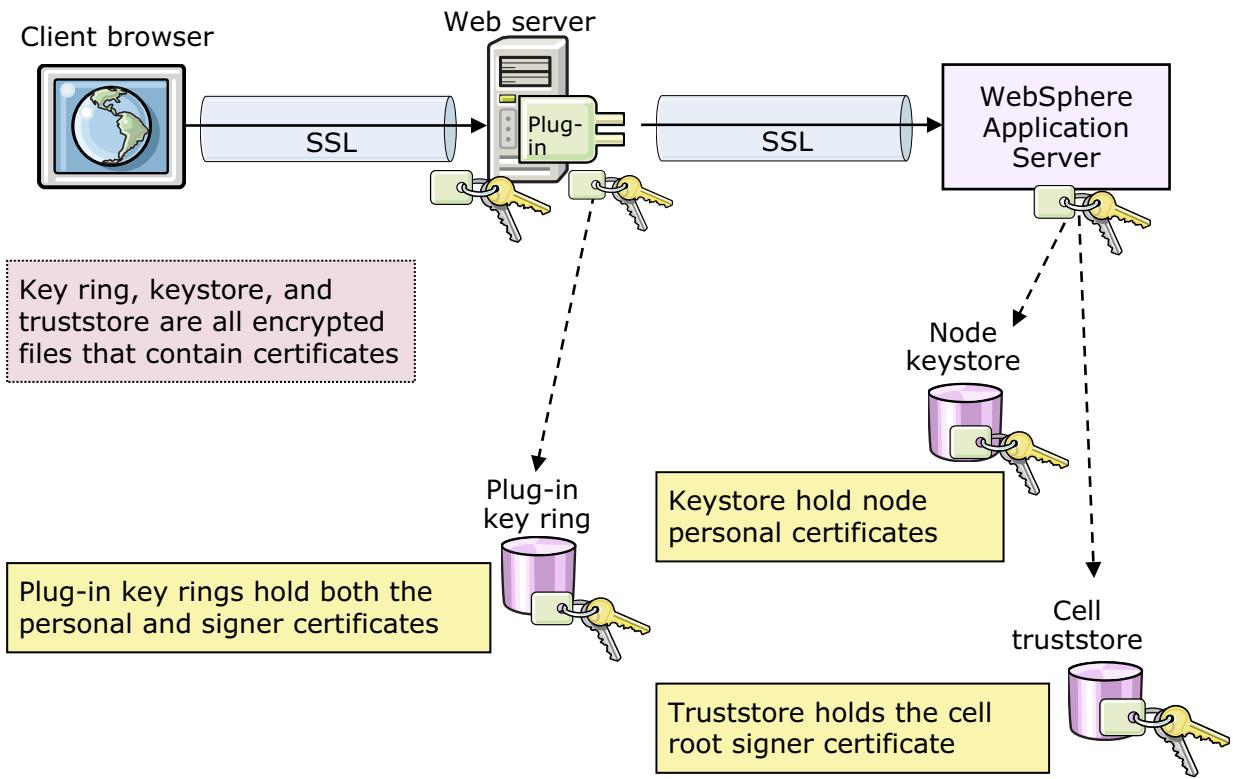
WA5913.0

Notes:

Because the client chooses its own session key, nobody else knows it. It can securely send that session key to the server with 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 that the certificate authority digitally signed.

What is the key ring, keystore, and a truststore?



© Copyright IBM Corporation 2013

Figure 11-27. What is the key ring, keystore, and a truststore?

WA5913.0

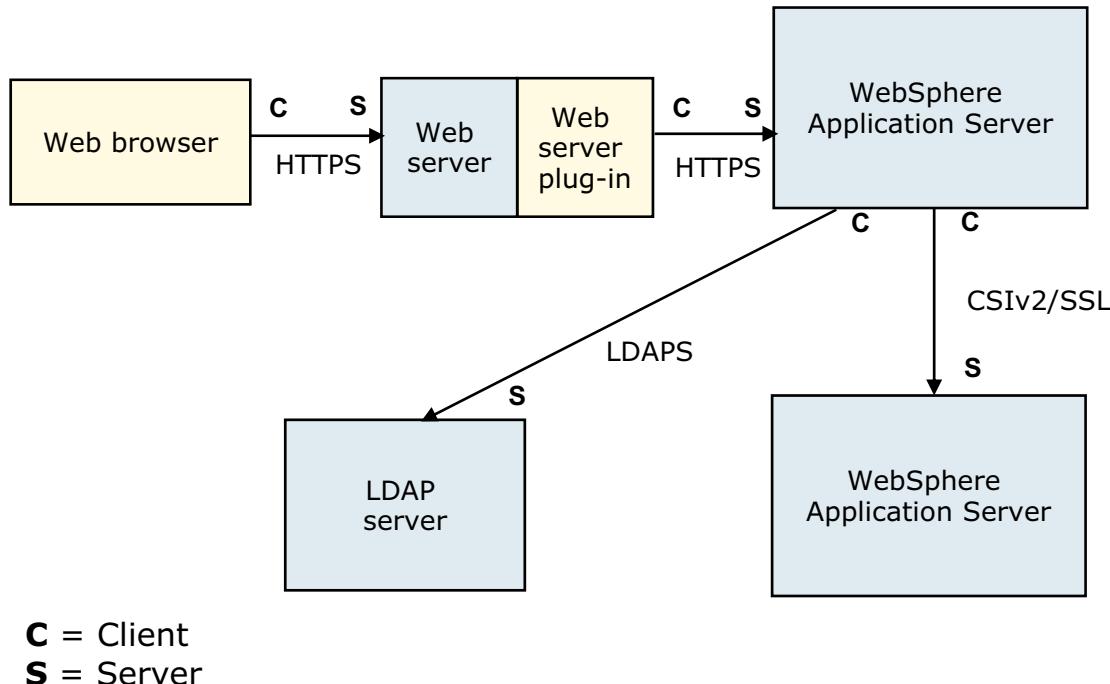
Notes:

The plug-in uses a single keystore (or key ring) which WebSphere generates for it. This file contains both the personal certificate for the plug-in as and the cell root signer.

The application servers (or nodes) use two different files. A keystore contains the node personal certificate, and the truststore holds the signer certificates the node chooses to trust.

By default, that means just the cell root signer certificate.

SSL client/server endpoint identification



© Copyright IBM Corporation 2013

Figure 11-28. SSL client/server endpoint identification

WA5913.0

Notes:

This diagram illustrates the various client/server endpoints in SSL communication between the user at the browser and the various points within WebSphere. Understanding who is the client and server in an SSL connection allows the administrator to properly configure the keystores, truststores, or other configuration.

Understand that “client,” in this context, can 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 might 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; it just verifies that a trusted signer signed the certificate. This process is different from what WebSphere does for a user certificate that is used for authentication, where it verifies that the user is in the registry.

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 suggested solution is to use socket factories, which define their own keystore or truststore, without using the system properties
- For an example, go to:
 - http://www.ibm.com/developerworks/websphere/techjournal/0612_birk/0612_birk.html

© Copyright IBM Corporation 2013

Figure 11-29. SSL problems: Handshake failures (1 of 3)

WA5913.0

Notes:

A **keystore** contains the personal certificates that can be used as the identity for the SSL endpoint that references the keystore. 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 can be exchanged. The personal certificate that the SSL configuration references and stores in the keystore is the certificate that is used.

A **personal certificate** represents the identity of the endpoint and contains a public and private key for signing or encrypting data.

A **truststore** contains the signer certificates that this endpoint trusts when either making connections (from an outbound endpoint) or accepting connections (for an inbound endpoint).

A **signer certificate** represents a certificate and public key that is associated with some personal certificate. The purpose of the signer certificate is to verify personal certificates. By accepting the signer certificate into the truststore of an endpoint, you are allowing the owner of the private key to establish connections with this endpoint. The signer certificate explicitly trusts connections that are made to or by the owner of the associated personal certificate. The signer certificate is typically made public by the owner of the personal certificate. It is up to the receiving entity to determine whether it is a trusted signer before adding it to the truststore.

Socket factories are a simple way to capture various policies that are related to the sockets that are constructed. Producing such sockets in a way that does not require special configuration of the code that asks for the sockets. Due to polymorphism of both factories and sockets, application code can use different kinds of sockets just by passing different factories.

Factories can themselves be customized with parameters used in socket construction. For example, factories might be customized to return sockets with different networking timeouts or security parameters already configured. The sockets that are 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.

If socket factories cannot be used, the following technote provides methods to resolve this situation:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg21191941>

SSL problems: Handshake failures (2 of 3)

javax.net.ssl.SSLHandshakeException: The client and server were not able to negotiate the wanted 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
 - Possibly 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

© Copyright IBM Corporation 2013

Figure 11-30. SSL problems: Handshake failures (2 of 3)

WA5913.0

Notes:

The `ssl.client.props` file, which is 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=
```

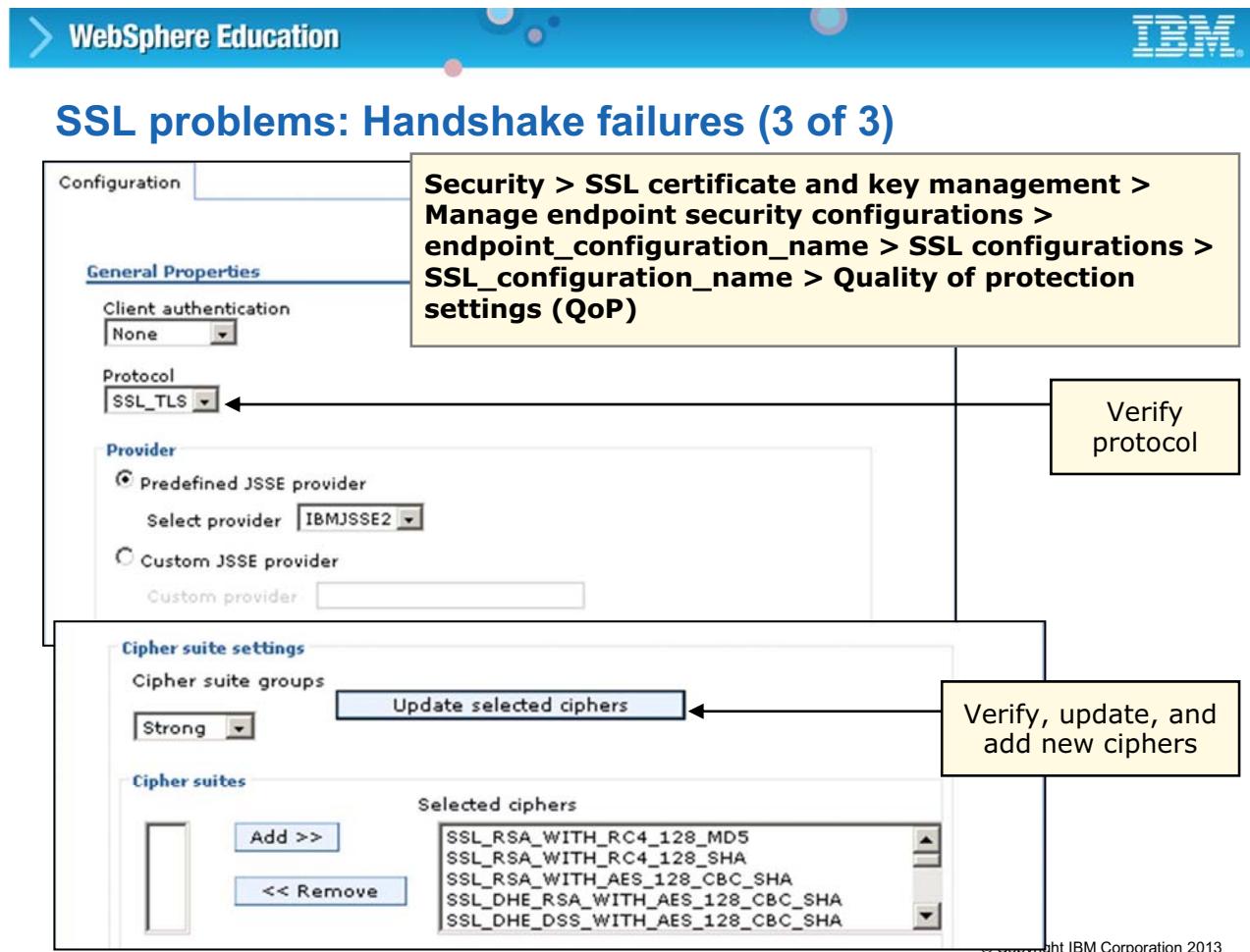


Figure 11-31. SSL problems: Handshake failures (3 of 3)

WA5913.0

Notes:

Client authentication specifies whether the 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 a client certificate to be sent during the handshake. If **Supported** is selected, the server requests a client certificate to be sent. If the client does not have a certificate, the handshake might still succeed. If **Required** is selected, the server requests a client certificate to 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.

Common SSL connection error message

- SSL handshake failure
 - Error when no trusted certificate is found
 - The certificate alias cannot be found in the truststore

WSX509TrustMa E CWPKI0022E: SSL HANDSHAKE FAILURE: A signer with SubjectDN "CN=was8host01, OU=was8host01Node01Cell, OU=was8host01Node02, O=IBM, C=US" was sent from target host:port "localhost:9049".

The signer may need to be added to local trust store "C:/Program Files/IBM/WebSphere/AppServer/profiles/profile1/config/cells/was8host01Cell01/trust.p12" located in SSL configuration alias "NodeDefaultSSLSettings" loaded from SSL configuration file "security.xml".

The extended error message from the SSL handshake exception is: "PKIX path building failed: java.security.cert.CertPathBuilderException: PKIXCertPathBuilderImpl could not build a valid CertPath.; internal cause is:

java.security.cert.CertPathValidatorException: The certificate issued by CN=was8host01, OU=Root Certificate, OU=was8host01Node01Cell, OU=was8host01Node02, O=IBM, C=US is not trusted; internal cause is:

java.security.cert.CertPathValidatorException: Certificate chaining error".

© Copyright IBM Corporation 2013

Figure 11-32. Common SSL connection error message

WA5913.0

Notes:

This message shows an example of one of the most common errors that occur when attempting to establish an SSL connection from a client to a server. In this case, the server sent a certificate that the client does not recognize. 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 truststore that is being used. This information 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** truststore specified.

A CWPKI0022E error message is usually followed by a CWPKI0428I message like:

CWPKI0428I: The signer might need to be added to the local trust store. You can use the **Retrieve from port** option in the administrative console to retrieve the certificate and resolve the problem. If you determine that the request is trusted, complete the following steps:

1. Log in to the administrative console.
2. Expand Security and click **SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations**.

3. Select the appropriate outbound configuration to get to the `(cell):was8host01Cell01` management scope.
4. Under Related Items, click **Key stores and certificates** and click the **CellDefaultTrustStore** key store.
5. Under Additional Properties, click **Signer certificates** and **Retrieve From Port**.
6. In the Host field, enter `localhost` in the host name field, `9049` in the Port field, and `localhost_cert` in the Alias field.
7. Click **Retrieve Signer Information**.
8. Verify that the certificate information is for a certificate that you can trust.
9. Click **Apply** and **Save**.

Steps to diagnose SSL handshake issues (1 of 3)

1. Identify the endpoints for the SSL communication
 - Determine who is the SSL client (client initiates the SSL handshake) and who is the SSL server (the receiving party in the SSL handshake attempt)
 - Sometimes not obvious, as the SSL client can be:
 - Java EE client or thin client
 - Web browser
 - WebSphere process
 - The SSL server is usually a WebSphere process
2. SSL handshake issues occur between two endpoints, and the SSL handshake error usually surfaces in the error log of the client
 - Determine the SSL configuration, keystore, and truststore for the SSL client and SSL server

© Copyright IBM Corporation 2013

Figure 11-33. Steps to diagnose SSL handshake issues (1 of 3)

WA5913.0

Notes:

1. Identify the endpoints for the SSL communication. It is important to determine who is the SSL client (the client that initiates the SSL handshake) and who is the SSL server (the receiving party in the SSL handshake attempt). This identity is sometimes not obvious, as the SSL client can be a Java Platform, Enterprise Edition client or thin client, a web browser, or a WebSphere process. The SSL server is usually a WebSphere process.
2. SSL handshake problems occur between two endpoints, and the SSL handshake error usually surfaces in the error log of the client. Determine the SSL configuration, keystore, and truststore for the SSL client and SSL server.
3. Knowledge of the certificates can be used to identify where the setup might be incorrect. Display the certificate information for the signer certificate and personal certificate (if there is one) using the administrative console SSL management, or iKeyman utility.
4. From the displayed information, check the validity period for the signer certificate on the SSL client side and SSL server side. Verify that the current date falls within the start date and end date for the certificate. The certificate's start date should precede the current date and not be expired.

5. From the displayed information, confirm that the SSL client's truststore contains the signer certificate of the server. Verify that the issuer's distinguished name and subject distinguished name for the server's signer certificate on the SSL client side match the server personal certificate on the SSL server side.

Steps to diagnose SSL handshake issues (2 of 3)

3. Knowledge of the certificates can be used to identify where the setup might be incorrect
 - Examine the certificate information for the signer certificate and personal certificate (if there is one) using one of the following tools:
 - Administrative console SSL management
 - iKeyman utility
4. Check the validity period for the signer certificate on the SSL client side and SSL server side
 - Verify that the current date falls within the start date and end date for the certificates
 - The certificate's start date should precede the current date and not be expired

© Copyright IBM Corporation 2013

Figure 11-34. Steps to diagnose SSL handshake issues (2 of 3)

WA5913.0

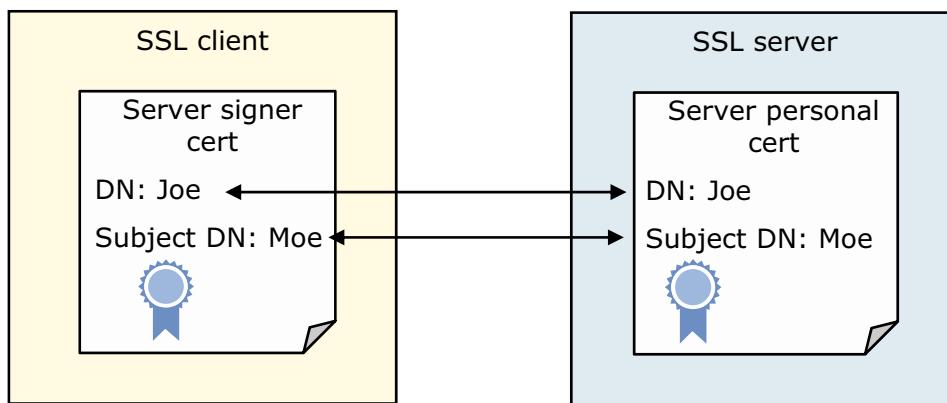
Notes:

z/OS notes:

- The keystore and truststore (repertoire) used most commonly on z/OS are the Security Products RACF (IBM), TSS, and ACF2 (CA).
- RACF TSO commands and product commands are used to verify the certificates instead of iKeyman on z/OS.

Steps to diagnose SSL handshake issues (3 of 3)

5. Confirm that the SSL client's truststore contains the signer certificate of the server
 - Verify that the issuer's distinguished name and subject distinguished name for the server's signer certificate on the SSL client side matches that of the server personal certificate on the SSL server side



© Copyright IBM Corporation 2013

Figure 11-35. Steps to diagnose SSL handshake issues (3 of 3)

WA5913.0

Notes:

Confirm that the SSL clients trust store contains the signer certificate of the server. Verify that the issuer's distinguished name and subject distinguished name for the server's signer certificate on the SSL client side match the server personal certificate on the SSL server side.

General SSL problem determination steps

- Check the `SystemOut.log` and `SystemErr.log` files to determine which SSL problem you are facing
- Use the search facility in the WebSphere Support site to look for common solutions
- If a common solution is not found, gather a JSSE trace
 - Set the following generic JVM arguments:
`-Djavax.net.debug=true -Djava.security.auth.debug=all`
 - Set the trace string `SSL=all`
 - Restart the application server
 - Re-create the problem and view the `SystemOut.log` and `trace.log` files
 - Look for exceptions and stack traces and work back up the thread to find the lines before the exception
 - The inputs like client key, server key, expiration dates, and ciphers are listed in the trace output

© Copyright IBM Corporation 2013

Figure 11-36. General SSL problem determination steps

WA5913.0

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.

11.4. Java 2 security problems

Java 2 security problems



© Copyright IBM Corporation 2013

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

8.0

Figure 11-37. Java 2 security problems

WA5913.0

Notes:

Java 2 security problems (1 of 2)

- Java 2 security access control exceptions at run time might result if:
 - An application is not prepared for Java 2 security
 - The application provider does not include a `was.policy` file as part of the application EAR file
 - 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:
 - Permission violation that causes the exception
 - Exception call stack
 - Permissions that are granted to each stack frame

© Copyright IBM Corporation 2013

Figure 11-38. Java 2 security problems (1 of 2)

WA5913.0

Notes:

The `was.policy` file is the template for application-specific permissions. The `was.policy` file is embedded in the enterprise archive (EAR) file.

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`?

Java 2 security problems (2 of 2)

- Handling Java 2 security access exceptions
 - Disable Java 2 security: Easy, but does organization security policies allow it?
 - Leave Java 2 security enabled, but then grant either:
 - The minimum required 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; };
```

© Copyright IBM Corporation 2013

Figure 11-39. Java 2 security problems (2 of 2)

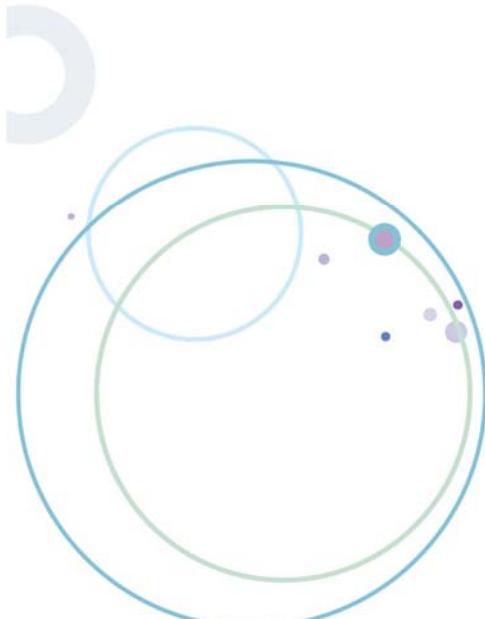
WA5913.0

Notes:

Here are some ways to handle Java 2 security access exceptions. You can disable Java 2 security, which is easy to do, but does your organization allow this policy? Alternatively, you can leave Java 2 security enabled, and then grant either enough extra permissions or all permissions to the problematic application. Use the PolicyTool to edit policy files and grant `java.security.AllPermission` in the WebSphere policy file that is embedded in the application EAR file. This slide shows an example policy file with all permissions granted.

11.5. More tools and techniques

More tools and techniques



© Copyright IBM Corporation 2013

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

8.0

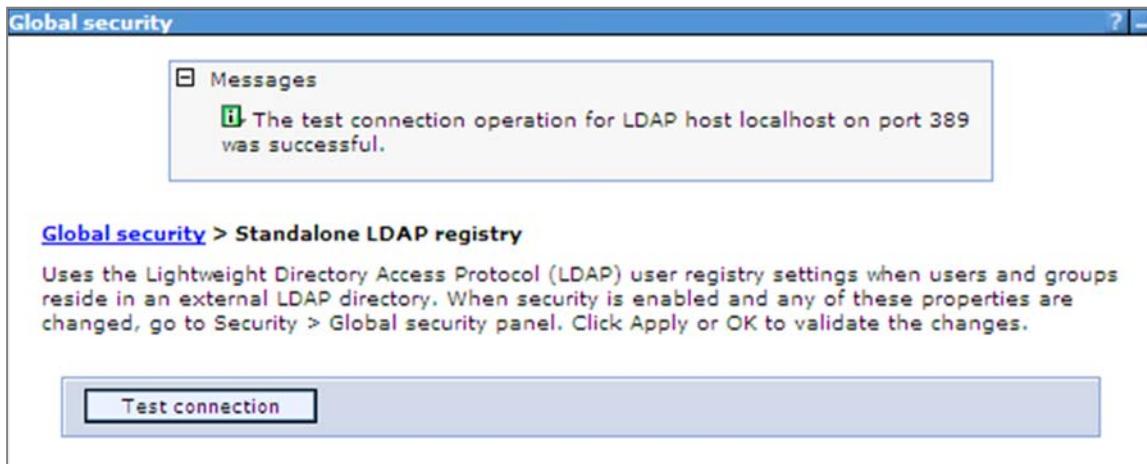
Figure 11-40. More tools and techniques

WA5913.0

Notes:

Administrative console security PD tools (1 of 3)

- The **Test connection** button attempts to connect to the LDAP server from the deployment manager by using the LDAP server host name and port



© Copyright IBM Corporation 2013

Figure 11-41. Administrative console security PD tools (1 of 3)

WA5913.0

Notes:

The Test connection link in the administrative console attempts to connect to the LDAP server from the deployment manager by using LDAP server host name and port.

Clicking the Test connection link is equivalent to running the following command:

```
AdminTask.validateLDAPConnection( [-type IBM_DIRECTORY_SERVER -hostname
<host_name> -port <your_port> -baseDN <your_base_DN> -bindDN <your_bind_DN>
-sslEnabled false ] )
```

In the lab environment for this course, the command would be as follows:

```
AdminTask.validateLDAPConnection( [-type IBM_DIRECTORY_SERVER -hostname localhost
-port 389 -baseDN dc=ibm,dc=com -bindDN uid=wsbind,cn=users,dc=ibm,dc=com
-sslEnabled false ] )
```



Administrative console security PD tools (2 of 3)

SSL certificate and key management

- Certificate expiration, endpoint SSL configuration, key managers, trust managers
- Keystores and certificates
 - Based on iKeyman functions from previous WebSphere versions

SSL certificate and key management > Key stores and certificates > CellDefaultKeyStore

Defines keystore types, including cryptography, RACF(R), CMS, Java(TM), and all truststore types.

General Properties		Additional Properties
Name	CellDefaultKeyStore	<ul style="list-style-type: none"> Signer certificates Personal certificates Personal certificate requests Custom properties
Description	Default key store for was7host01Cell01	
Management scope	(cell):was7host01Cell01	
Path	\${CONFIG_ROOT}/cells/was7host01Cell01/key.p12	

© Copyright IBM Corporation 2013

Figure 11-42. Administrative console security PD tools (2 of 3)

WA5913.0

Notes:

Security configuration report:

- Starts a security configuration report that displays the core security settings of the application server
- Also displays the administrative users and groups and the CORBA naming roles

A current limitation to the report is that it does not display application level security information. The report also does not display information on Java Message Service (JMS) security, bus security, or web services security.

When multiple security domains are configured, the report displays the security configuration that is associated with each domain.



Administrative console security PD tools (3 of 3)

- Security Configuration Report button
 - Generates a security configuration report that shows the core security settings of the application server
 - Also shows the administrative users and groups and the CORBA naming roles



© Copyright IBM Corporation 2013

Figure 11-43. Administrative console security PD tools (3 of 3)

WA5913.0

Notes:

The security configuration report link starts the security configuration report and displays the core security settings of the application server. It also displays the administrative users and groups in the CORBA naming roles.

Security configuration files (1 of 3)

security.xml

- Each profile has a copy of `security.xml` of the deployment manager 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 might override the cell-wide configuration
 - Enable or disable application security
 - Enable or 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

© Copyright IBM Corporation 2013

Figure 11-44. Security configuration files (1 of 3)

WA5913.0

Notes:

A server copy of `security.xml` is 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 that used to be specified in this file are now configured in `security.xml`.
- `soap.client.props` – To avoid having to expose user ID and password information on the command line when running the `wsadmin` tool, set the appropriate values for `com.ibm.SOAP.loginUserId=<userid>` and `com.ibm.SOAP.loginPassword=<passwd>`.

Security configuration files (2 of 3)

sas.client.props and **soap.client.props**

- Each profile has a copy of `sas.clients.props` in its properties directory
- `Soap.client.props` is used to store administrator ID and password

app.policy and **was.policy**

- Contains 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

wimconfig.xml

- Each profile has a copy of `wimconfig.xml` 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

© Copyright IBM Corporation 2013

Figure 11-45. Security configuration files (2 of 3)

WA5913.0

Notes:

Two important properties files, `sas.client.props` and `soap.client.props`, can contain administrative user IDs and passwords. You might need to manually update the passwords periodically. The `app.policy` and `was.policy` files contain permissions for Java 2 security. The `wimconfig.xml` and `wim` files contain virtual member manager configuration.



Security configuration files (3 of 3)

fileRegistry.xml

- If file-based user registry is configured, each profile has a copy of `fileRegistry.xml` at
`<WAS_HOME>\profiles\<Profile_Name>\config\cells\<Cell_Name>`
- This XML is the file repository and it contains:
 - User and group identifiers
 - Encrypted passwords for users
 - Passwords are encrypted by using a one-way hash and applying the message digest algorithm (specified in `wimconfig.xml`)

© Copyright IBM Corporation 2013

Figure 11-46. Security configuration files (3 of 3)

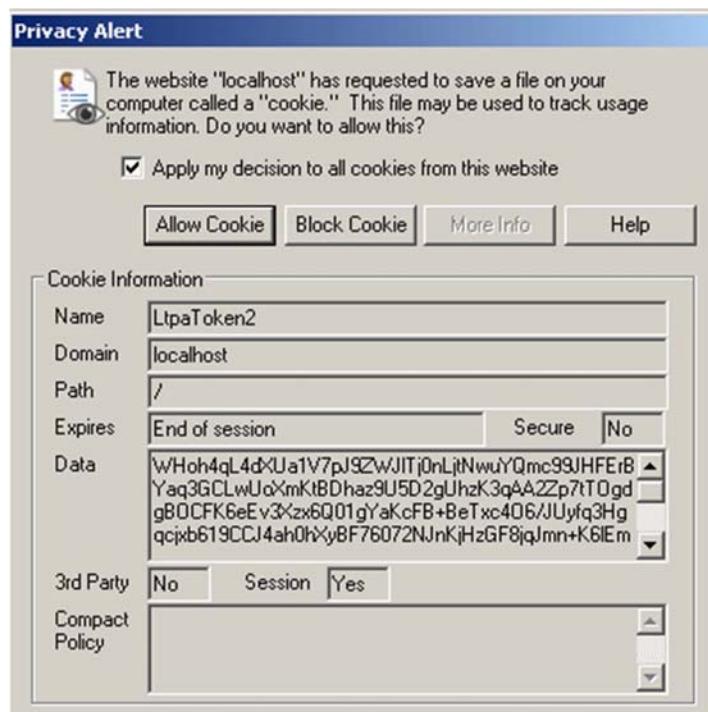
WA5913.0

Notes:

In the `fileRegistry.xml` file, the passwords are encrypted with a one-way hash by applying the message digest algorithm that is specified in the VMM configuration file (`wimconfig.xml`). The default value for the message digest algorithm is SHA-1. The `wsadmin` command `$AdminTask updateIdMgrFileRepository` can change the algorithm.

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 might have problems with:
 - Domain suffix for the LTPA SSO configuration
 - Proxy server configuration



© Copyright IBM Corporation 2013

Figure 11-47. Tracking LTPA tokens

WA5913.0

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**.

Similarly, you can configure a Mozilla Firefox browser to warn about cookies. Select **Tools > Options > Privacy**. Check the options for accepting cookies, and select the keep until option: **ask me every time**.

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. Your browser informs 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 probably swallowed the cookie. It might 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 whether the LTPA token is generated by WebSphere Application Server and then lost.

Known problems on interactions between security proxies and LTPA

1. If you log in to a proxy such as Tivoli Access Manager, it asserts an identity to WebSphere, which creates an LTPA cookie. If you later log in to Tivoli Access Manager from that same browser as a different user, and the LTPA cookie is not expired, you are one user to Tivoli Access Manager, and a different user to WebSphere. The fact that the LTPA exists and is valid prevents the TAI from being called and asserting the new identity.
2. Another LTPA problem 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 is removed. If the new user then goes to the application in the other domain, that user is logged in as the other user. With this problem, a security proxy hides that fact that multiple domains exist.

Disabling administrative security

- Sometimes it is necessary to disable administrative security to troubleshoot security-related problems
- If the application server or deployment manager is running, use the administrative console
 - Select **Security > Secure administration**
 - Clear **Enable administrative security**
 - Save changes to the master repository
- If the application server cannot be started, for example the password of the server user ID in the user registry is expired or the user registry cannot be reached for authentication
 - Disable administrative security by using the command line:


```
<WAS_HOME>\bin\wsadmin.bat -conntype NONE
wsadmin>securityoff
wsadmin>quit
Start server1 or dmgr
```
 - **Note:** Changes only the `security.xml` file of the local node

© Copyright IBM Corporation 2013

Figure 11-48. Disabling administrative security

WA5913.0

Notes:

This procedure should work without any problem. However, in case it fails, you might 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.

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 tracing on relevant security components
- Analyze and interpret trace information
- Recognize symptoms of common Secure Sockets Layer (SSL) configuration problems
- Recognize symptoms of common Java 2 security problems
- Locate the security configuration files
- Use tools to validate the security configuration files
- Use wsadmin securityoff to disable security

© Copyright IBM Corporation 2013

Figure 11-49. Unit summary

WA5913.0

Notes:



Checkpoint questions

1. Which two application server components have a security collaborator process?
2. In which log files would you most likely find stack traces that result 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?

© Copyright IBM Corporation 2013

Figure 11-50. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

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

Checkpoint answers

1. The web container and the EJB container each have a security collaborator.
2. The SystemOut.log and SystemErr.log would contain stack traces that result from security-related exceptions.
3. For each application server, node agent, and deployment manager, you can configure tracing of security components at different levels of detail. This information can be written to a trace.log file.
4. The security.xml file contains all of the global security configuration information.

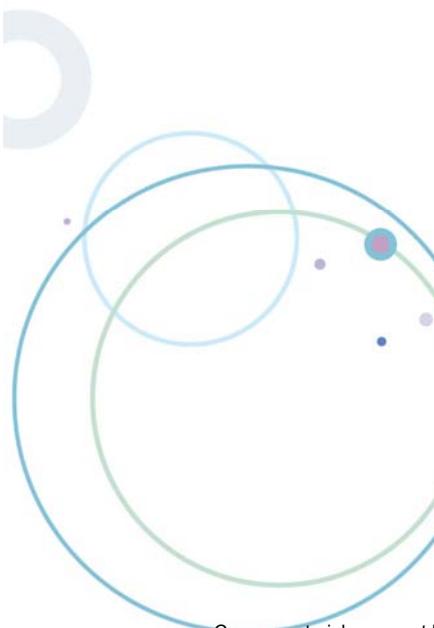
© Copyright IBM Corporation 2013

Figure 11-51. Checkpoint answers

WA5913.0

Notes:

Exercise 9



© Copyright IBM Corporation 2013

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

8.0

Figure 11-52. Exercise 9

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Detect security problems
- Examine and analyze log files for security messages
- Configure tracing on relevant WebSphere security components
- Examine and analyze trace logs
- Identify the security problems and fix them
- Analyze an SSL handshake failure
- Use wsadmin securityoff to disable security s

© Copyright IBM Corporation 2013

Figure 11-53. Exercise objectives

WA5913.0

Notes:

Unit 12. Application deployment problems

What this unit is about

This unit looks at common problems that can arise while deploying an application.

What you should be able to do

After completing this unit, you should be able to:

- Identify common problems with application installation
- Interpret runtime console messages and log messages
- Use application deployment troubleshooting tools
- Enable tracing of the appropriate components and interpret the data
- Use wsadmin to test and reproduce a problem
- Use the IBM Assembly and Deploy Tools for WebSphere Administration to validate application configuration files
- Use the Classloader Viewer tool to troubleshoot class loading issues
- Identify and troubleshoot class loader problems

How you will check your progress

- Checkpoint

References

WebSphere 8.0 Information Center

<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp>

Unit objectives

After completing this unit, you should be able to:

- Identify common problems with application installation
- Interpret runtime console messages and log messages
- Use application deployment troubleshooting tools
- Enable tracing of the appropriate components and interpret the data
- Use wsadmin to test and reproduce a problem
- Use the IBM Assembly and Deploy Tools for WebSphere Administration to validate application configuration files
- Use the Classloader Viewer tool to troubleshoot class loading issues
- Identify and troubleshoot class loader problems

© Copyright IBM Corporation 2013

Figure 12-1. Unit objectives

WA5913.0

Notes:

The value of WebSphere Application Server V8.0 is in the applications that it hosts. Therefore, the successful deployment of an application is of paramount importance. WebSphere Application Server V8.0 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. The *wsadmin* function is usually used by a WebSphere Application Server V8.0 administrator, a WebSphere Application Server V8.0 operator, an application developer, or a systems integration specialist (in most organizations, a single person has more than one of these roles).

Typically, an application is developed and tested in a WebSphere Application Server V8.0 test environment before 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, and proxies, are not 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.



Topics

- Application installation problems
- Troubleshoot class loader problems

© Copyright IBM Corporation 2013

Figure 12-2. Topics

WA5913.0

Notes:

12.1.Application installation problems

Application installation problems



© Copyright IBM Corporation 2013

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

8.0

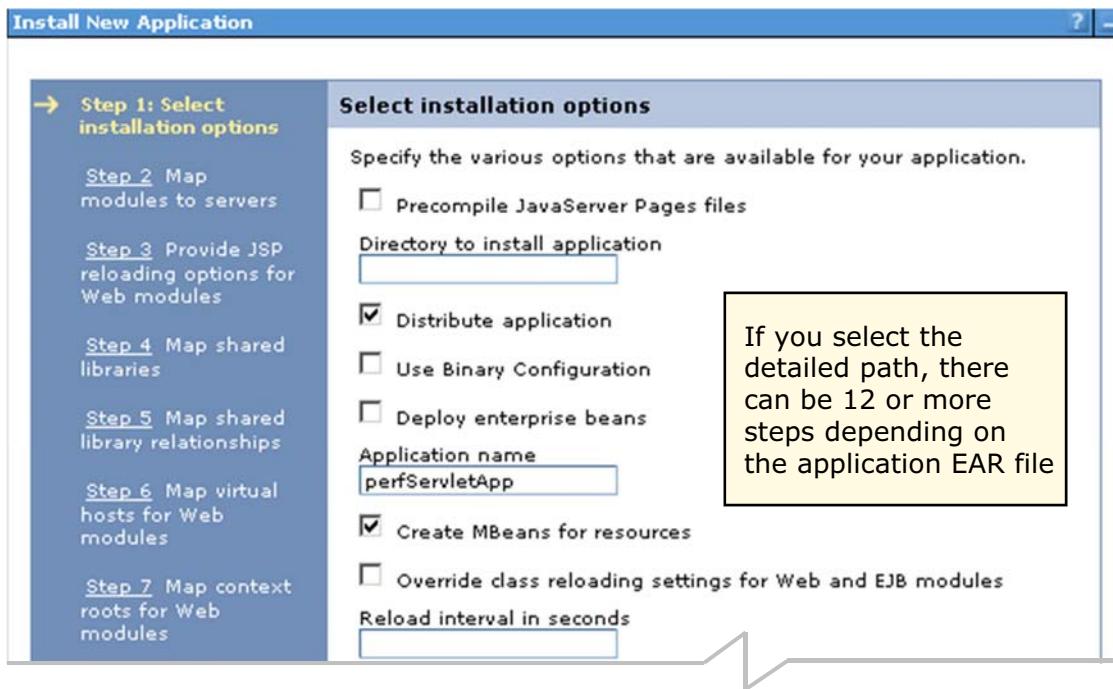
Figure 12-3. Application installation problems

WA5913.0

Notes:

Application installation methods: Console wizard

- Select Application > New Application > New Enterprise Application



© Copyright IBM Corporation 2013

Figure 12-4. Application installation methods: Console wizard

WA5913.0

Notes:

An application can be installed into WebSphere by using two primary methods:

1. The administrative console installation wizard, by navigating through **Applications > Enterprise Applications > Install Applications > Install New Application**
2. wsadmin script interpreter
`wsadmin $AdminApp.install('<file_name>.ear')`

The administrative console and wsadmin can also be used to update running applications.

Uninstall the old EAR file and install the updated EAR file.

An application deployment issue can manifest itself as:

- A warning during the deployment process
- An error while the application information is being persisted to the WebSphere Application Server V8.0 configuration files
- A failed request when the application is tested

In any of these cases the result is the same: the application is not going to work.

During the installation of the application, exceptions can occur as a result of a failure by the installation process. The application is installed, and failure can occur while trying to start the application, or after the application is started and an actual transaction or request is sent to the application.

The WebSphere Application Server V8.0 Information Center is an excellent reference for the material explained in this unit.

Application installation methods: wsadmin

- `wsadmin` can run interactively or it can run installation scripts from the command line
- To run interactively, use the command:
 - `wsadmin $AdminApp.installInteractive('app.ear')`
- To accept all default options (fast path), use the command:
 - `wsadmin $AdminApp.install('app.ear')`
- To run and installation script, use the command:
 - `wsadmin -f <jython file>`
- Success message:

ADMA5013I: Application BadAppEAR installed successfully

© Copyright IBM Corporation 2013

Figure 12-5. Application installation methods: wsadmin

WA5913.0

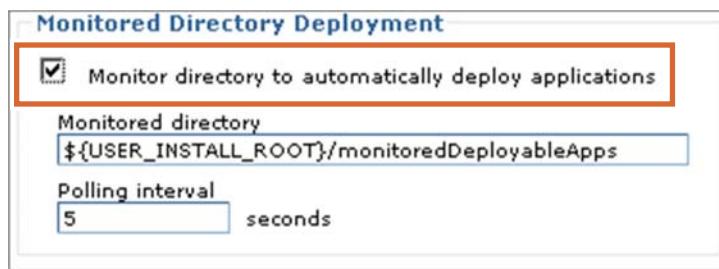
Notes:

The steps in this task return a success message if the system successfully installs the application. When installing large applications, the command might return a success message before the system extracts each binary file. You cannot start the application until the system extracts all binary files. If you installed a large application, use the `isAppReady` and `getDeployStatus` commands for the `AdminApp` object to verify that the system extracted the binary files before starting the application.

The `isAppReady` command returns a value of true if the system is ready to start the application, or a value of false if the system is not ready to start the application. For example, use Jython: `print AdminApp.isAppReady('application1')`

Application installation methods: Monitored directory

- You can install an enterprise application file on an application server by dragging or copying the archive to a monitored directory
- The supported archives are:
 - Enterprise application archive (EAR)
 - Web application archive (WAR)
 - Java archive (JAR)
 - Session Initiation Protocol (SIP) archive (SAR)
- An enterprise application file must conform to the Java Platform, Enterprise Edition (Java EE) specification
- By default, monitored directory deployment is not enabled



© Copyright IBM Corporation 2013

Figure 12-6. Application installation methods: Monitored directory

WA5913.0

Notes:

New feature in version 8: You can deploy an EAR, JAR, WAR, or SAR file to an application server by dragging or copying the file to a monitored directory. For base (stand-alone) application servers, the monitored directory is the `monitoredDeployableApps/servers/server_name` directory of the application server profile. The product scans a monitored directory for new applications no more frequently than every 5 seconds, by default. After finding a new EAR, JAR, WAR, or SAR file in a monitored directory, the product installs the file on the application server and starts the application or module.`newfeat`.

After you add an EAR file to a monitored directory, the product creates a temporary copy of the EAR file in another directory and installs the file on the server. After you add a JAR, WAR, or SAR file to a monitored directory, the product creates a temporary copy of the archive in another directory. It wraps the archive in an EAR file that is named `archive_extension.ear`, and installs the new EAR file. For example, `simpleApp.war` is installed as `simpleApp_war.ear`. The original archive that you added to the monitored directory is not changed.

Problem areas in application installation

During the application deployment process, you might encounter problems while doing any of the following tasks:

- Installing 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 messages
 - Namespace problems
- Application does not work as expected
 - Data source and connection factory problems
 - Resource connectivity issues
 - Namespace and scoping issues
 - Application security issues: authorization failures

© Copyright IBM Corporation 2013

Figure 12-7. Problem areas in application installation

WA5913.0

Notes:

During application deployment, configuration values such as JNDI names and security information might need to be provided. If any of the specified values are incorrect, they might not be detected until the application is started or run for the first time. This result is also true if incorrect credentials or method permissions are specified. They are not detected until the resource they protect is accessed.

If any problems are detected during the deployment process, then one of the following occurs:

- An error prevents the application from being deployed.
- A warning does not prevent the application from being deployed but might require correction before the application runs correctly.

However, the absence of an error or warning does not guarantee that the application is going to function correctly. As stated, some errors are not detected until the application is started or run for the first time.

Examining console and error log file messages

- During the installation process, any exception that occurs is logged to the `SystemErr.log` file
- If you are using the application installation wizard of the administrative console, the errors are displayed to the administrative console
 - For example, the following error message is a result of a corrupted or incorrect application EAR file:



© Copyright IBM Corporation 2013

Figure 12-8. Examining console and error log file messages

WA5913.0

Notes:

Application deployment problems, like all WebSphere Application Server V8.0 errors, warnings, and information messages are logged to the system logs based on the level of trace that is specified for the system. By default, any error that occurs is captured in the `SystemErr.log` or the `SystemOut.log` (or both). If you are using the administrative console, an error is rendered to the panel that is 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 are 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, by using the *wsadmin* script interpreter. However, when you look for your application in the administrative console, it does not show up because the *save* command was not run after the installation process. As more experience is gained with the WebSphere Application Server V8.0 installation process, user error occurs less frequently. Automating the installation process with scripts is the best way to reduce human errors.



wsadmin command-line exceptions

- Installing an EAR file from the command line using wsadmin might result in the following exceptions:

```
C:\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

wsadmin>$AdminApp install PlantsByWebSpherePD-Bad.ear
WASX7015E: Exception running command: "$AdminApp install PlantsByWebSpherePD-Bad.ear"; exception information:

com.ibm.wsspi.container.ContainerException: []
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 seems to be missing in the EAR file

© Copyright IBM Corporation 2013

Figure 12-9. wsadmin command-line exceptions

WA5913.0

Notes:

As with many other troubleshooting activities, the error is in the log. The difficulty is sometimes in recognizing the root cause of the error. In the example here, the error circled in red indicates that the EAR file is “bad.” This indication is not a moral judgment, nor is it the root of the problem. By reading further, it is seen that there is no deployment descriptor. Both of these indications fit in nicely within a small section of text. Often the symptomatic error and the root cause are much further apart in the log file.

An error message that results while running the wsadmin command

`$AdminApp.installInteractive()` or `$AdminApp.install()` has the identifier WASX7015E.

Typical application installation errors

- While installing 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 2013

Figure 12-10. Typical application installation errors

WA5913.0

Notes:

These errors are runtime exceptions that are based on the failure of the installation tool as it attempts to install the application, not on the runtime of the application itself. So while the reported symptom 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. To put it in a simple way, an application that is not installed “cleanly” does not run. If the installation issue goes unnoticed, it is caught when the application is run.

Any exception that occurs is logged to the `SystemErr.log` file. Some of the exceptions or errors have message identifiers with them. For example, error messages while running the `wsadmin` command `$AdminApp.installInteractive()` or `$AdminApp.install()` have the identifier `WASX7015E`. This identifier can be traced through the information center for its exact meaning by doing the following steps:

1. Navigate to the information center library at:
<http://www.ibm.com/software/webservers/appserv/was/library/>
2. Select the appropriate information center for your WebSphere Application Server V8.0 product.

3. When the information center is open, search on the identifier `WASX7015E` by entering it in the search text field near the top and clicking **Go**.
4. Look through the search result that best fits your query. With this identifier, it is the 100% result page. The identifier is highlighted wherever it is found in the results.

Typical application startup errors

- The installation process might 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 are:
 - `java.lang.ClassNotFoundException: class name Bean_AdderServiceHome_04f0e027Bean`
 - `ConnectionFac 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`
- For other common errors, see the information center topic: *Application startup problems*

© Copyright IBM Corporation 2013

Figure 12-11. Typical application startup errors

WA5913.0

Notes:

See the information center topic *Application startup problems*:

http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/rtrb_appstart.html

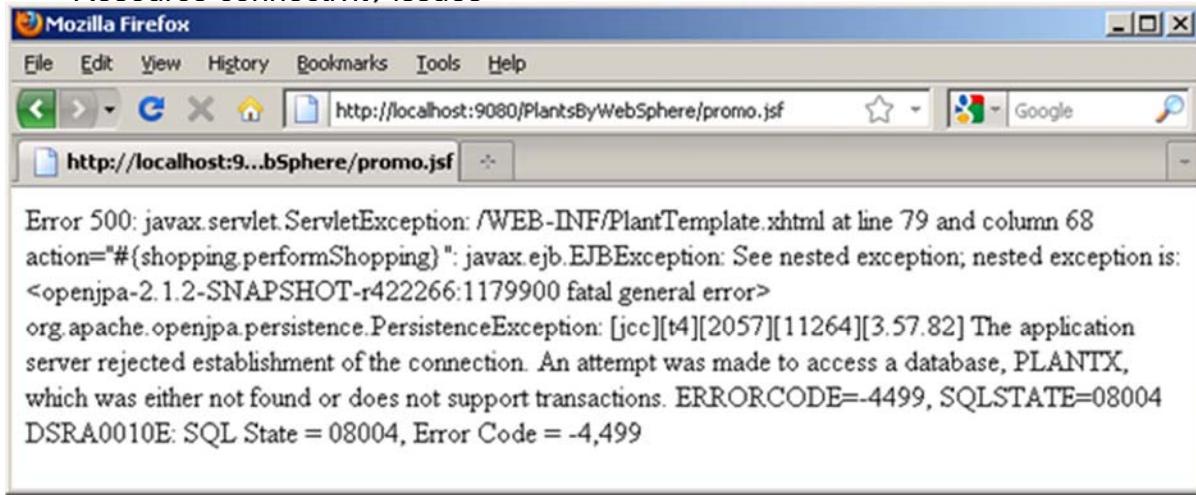
After an application is installed for the first time, its initial state is *stopped*. An explicit start is required to bring the application online. Depending on the environment topology, such as whether the application was deployed into a cluster, the actual *start* operation might vary. There might be a single instance of the application, or many instances that start simultaneously, 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 are satisfied but now that the runtime bindings are being exercised for the first time, some reference is not satisfied. You can tell by the types of errors that are detected, such as class loader exceptions, versioning problems, and naming reference issues.

 IBM

Typical application first-run errors (1 of 2)

- Running the first transaction through the application might result in issues such as:
 - javax.naming.NameNotFoundException for unresolved class path problems
 - Scoping issues
 - Resource connectivity issues



© Copyright IBM Corporation 2013

Figure 12-12. Typical application first-run errors (1 of 2)

WA5913.0

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 contains 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 that is returned to the browser. However, transaction failure can be more subtle. Your initial request might result in the return of content or results, but these results might be faulty. Maybe a graphic is missing because a relative path is incorrect, or a JSP that must be created on every request has an error that prevents compilation. Then, the various browser differences must be considered. Depending on the client that accesses the content, the type of client that is used can have a significant impact on what is considered a correct operation. Having your application work perfectly with Firefox but not with other browsers quickly results in a defect report from your users.



Typical application first-run errors (2 of 2)

Address <http://localhost:9080/PlantsByWebSphere/>

PLANTS BY WEBSPHERE

Flowers Fruits & Vegetables Trees Accessories HOME : SHOPPING CART :

An Error has occurred during PlantsByWebSphere processing
Jsp Error Page

Using attributes javax.servlet.error.message ...status_code ...exception as specified by Servlet 2.2 to get information

Processing request:<http://localhost:9080/PlantsByWebSphere/error.jsp>
StatusCode: 500

Message:CORBA TRANSACTION_ROLLEDBACK 0x0 No; nested exception is:
org.omg.CORBA.TRANSACTION_ROLLEDBACK: javax.transaction.TransactionRolledbackException;
nested exception is: com.ibm.ws.ejb.persistence.utilpm.PersistenceManagerException: PMGR1014E: Exception occurred when getting connection factory: com.ibm.websphere.naming.CannotInstantiateObjectException:
Exception occurred while the JNDI NamingManager was processing a javax.naming.Reference object. [Root exception is com.ibm.websphere.naming.CannotInstantiateObjectException: Exception occurred while the JNDI NamingManager was processing a javax.naming.Reference object. [Root exception is java.lang.reflect.InvocationTargetException]] vmcid: 0x0 minor code: 0 completed: No
Exception:javax.transaction.TransactionRolledbackException: CORBA TRANSACTION_ROLLEDBACK

© Copyright IBM Corporation 2013

Figure 12-13. Typical application first-run errors (2 of 2)

WA5913.0

Notes:

Applications can include JSP error pages to show friendly error messages to the user. However, in development and test environments, it is more useful to display as much detail about the error as possible. The screen capture on this slide provides plenty of diagnostic data that is helpful for troubleshooting the problem such as error codes, exceptions, and stack trace.



Deployment troubleshooting tools (1 of 2)

- The Class Loader Viewer is embedded in the **troubleshooting** section of the administrative console
 - Class loader viewer service must be enabled for an application server

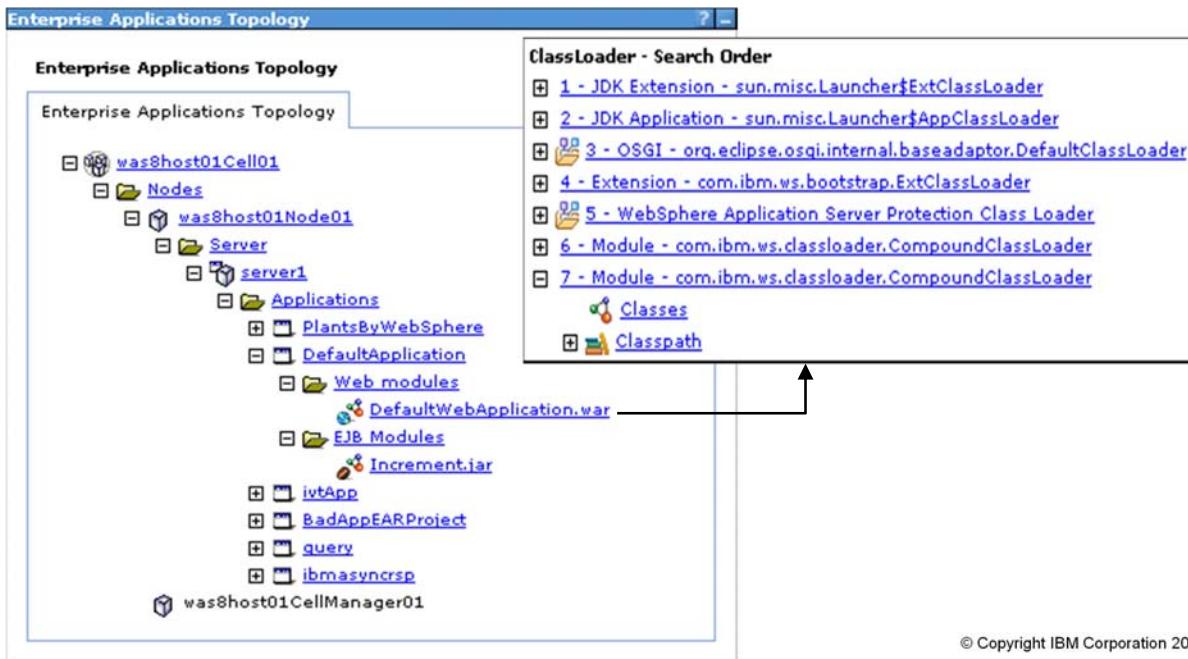


Figure 12-14. Deployment troubleshooting tools (1 of 2)

WA5913.0

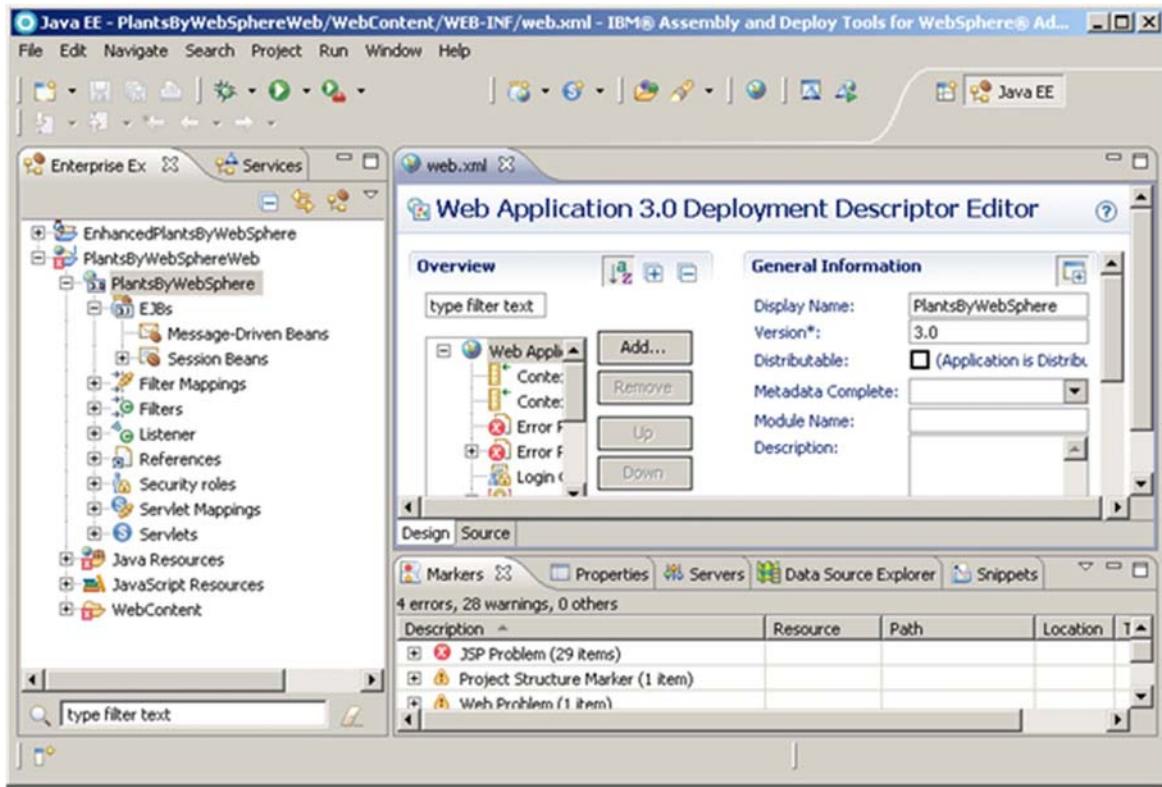
Notes:

Class loading is a critical part of the application server environment. The order of precedence of the class loaders is responsible for coexistence between the services that WebSphere Application Server V8.0 provides and the application functionality. As of version 6.0.2 of 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 Viewer* provides greatly facilitate resolution of class loader problems.

The WebSphere Application Server V8.0 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 V8.0 environment provides the services that the application uses 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 V8.0 containers.



Deployment troubleshooting tools (2 of 2)



© Copyright IBM Corporation 2013

Figure 12-15. Deployment troubleshooting tools (2 of 2)

WA5913.0

Notes:

With IBM Assembly and Deploy Tools for WebSphere Administration, you can assemble and deploy applications to run on WebSphere Application Server, Version 8.0. Using the extensible and easily customized workbench, you can quickly assemble, test, and deploy web, Java, and Java EE applications.

Assembly and deployment features include:

- Fully integrated tools and support for IBM WebSphere Application Server Version 8.0:
 - Tools for testing, debugging and publishing code
 - Automated deployment descriptor generation and visual editors
 - Jython scripting editor and debugger
 - SIP application support
- Tools, including many simple wizards and visual editors to help with the assembly and deployment of applications that use the Java Platform, Enterprise Edition (Java EE) and Java 2 Enterprise Edition. Java EE programming models include applications that use Java, web services, Enterprise JavaBeans (EJB), and Java Persistence API (JPA):

- JavaServer Faces tools to create a web user interface
- JAX-RPC web service import, creation, discovery, deployment, and testing
- JAX-WS web service deployment; policy set association and binding
- XML creation and validation
- XML form-based deployment descriptor editors
- Tools for adding database access to your applications, including built-in support and JDBC drivers for many widely used databases:
 - Tools for building JSR 168 portlets
 - Extensible and easily customized workbench

Gathering trace data for deployment problems

- From the administration console, enable diagnostic trace on the specific application server by using the following trace string

```
*=info:com.ibm.ws.management.*=all:com.ibm.websphere.management.*=all
```

- If the deployment is done by using wsadmin, enable wsadmin tracing

- Edit `wsadmin.properties` file that is located under the `<profile_root>/properties` folder
- Uncomment the following line:

```
#com.ibm.ws.scripting.traceString=com.ibm.*=all=enabled
```

- Enable the following trace on the node agent and Application Server (when using wsadmin script):

```
*=info:com.ibm.ws.management.*=all:com.ibm.websphere.management.*=all
```

© Copyright IBM Corporation 2013

Figure 12-16. Gathering trace data for deployment problems

WA5913.0

Notes:

- In the administrative console, navigate to **Troubleshooting > Logs and Trace > dmgr > Diagnostic Trace > Change Log Detail Levels**.

If you are in a Base environment, select **server1** instead of the **dmgr** server.

- Select the **Runtime** tab.

- 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 that is created outside of `<WAS_HOME>` directory.

12.2.Troubleshooting class loader problems

Troubleshooting class loader problems



© Copyright IBM Corporation 2013

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

8.0

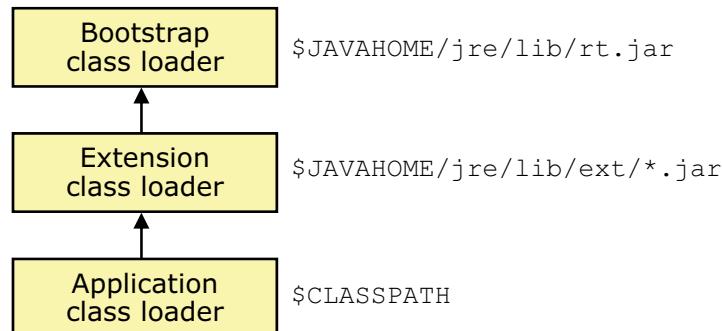
Figure 12-17. Troubleshooting class loader problems

WA5913.0

Notes:

Class loading concepts (1 of 2)

- Java class loaders hierarchy



- When the application class loader must load a class
 - It first delegates to the extensions class loader
 - Which, in turn, delegates to the bootstrap class loader
- If the parent class loader cannot load the class
 - The child class loader tries to find the class in its own repository
- In this manner, a class loader is only responsible for loading classes that its ancestors cannot load

© Copyright IBM Corporation 2013

Figure 12-18. Class loading concepts (1 of 2)

WA5913.0

Notes:

If the application class loader needs to load a class, it first delegates to the extensions class loader, which, in turn, delegates to the bootstrap class loader. If the parent class loader cannot load the class, the child class loader tries to find the class in its own repository. In this manner, a class loader is only responsible for loading classes that its ancestors cannot load.

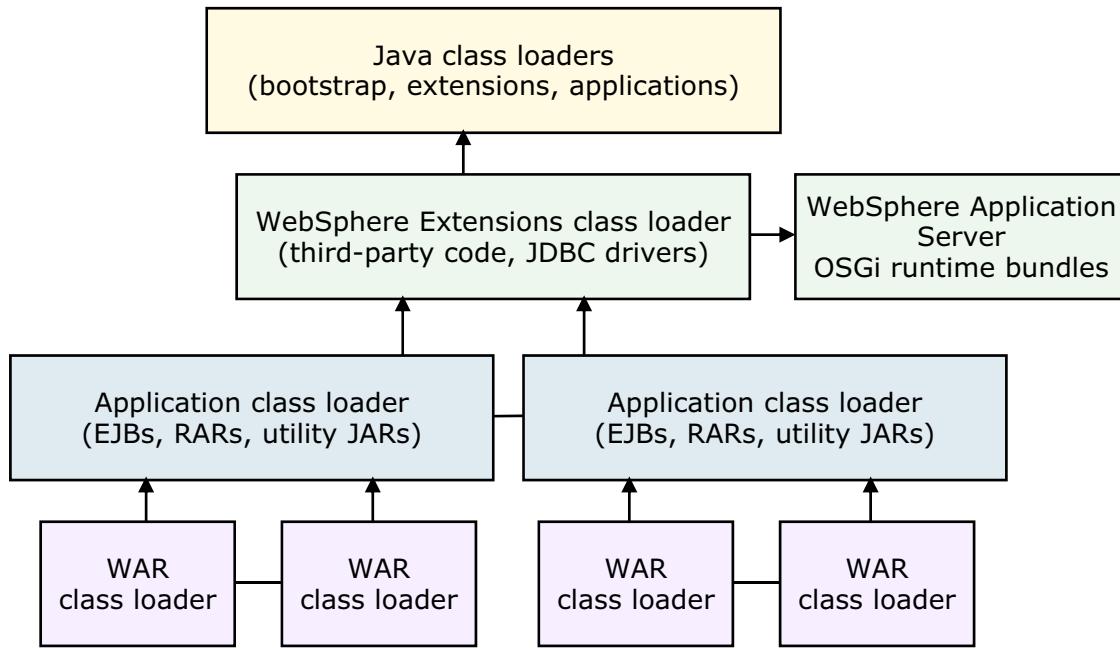
The hierarchy of class loaders that WebSphere uses, from highest to lowest, is as follows:

- Java class loaders
- WebSphere extension class loaders
- Application module class loader
- Web module class loader

Three parameters control class loading:

- Class loader mode
- War class loader policy
- Isolation policy

Class loading concepts (2 of 2)



© Copyright IBM Corporation 2013

Figure 12-19. Class loading concepts (2 of 2)

WA5913.0

Notes:

The top box represents the Java (bootstrap, extensions, and application) class loaders. WebSphere loads the minimum here to get itself bootstrapped and to initialize the WebSphere extensions class loader.

Application and web module class loaders

Java EE applications consist of five primary elements: web modules, EJB modules, application client modules, resource adapters (RAR files), and utility JAR files. Utility JAR files contain code that both EJBs and servlets can use. Utility frameworks such as log4j are good examples of a utility JAR file.

EJB modules, utility JAR files, resource adapter files, and shared libraries that are associated with an application are always grouped into the same class loader. This class loader is called the application class loader. Depending on the class loader policy, multiple applications (EAR files) can share this class loader, or it can be unique for each application, which is the default.

By default, web modules receive their own class loader, a WAR class loader, to load the contents of the WEB-INF/classes and WEB-INF/lib directories. You can modify the default behavior by changing the application's WAR class loader policy. The default is to have a class loader for each

WAR file in the application (this setting was called “Module” in previous releases). If the WAR class loader policy is set to “Single class loader for application” (called “Application” in previous releases), the application class loader loads the web module contents. The EJB files, RAR files, utility JAR files, and shared libraries are also loaded. The application class loader is the parent of the WAR class loader.

The application and the WAR class loaders are reloadable class loaders. They monitor changes in the application code to automatically reload modified classes.

You can modify this behavior at deployment time.

Handling JNI code

A JVM has only a single address space, and native code can be loaded only one time per address space. As a result, the JVM specification states that only one class loader in a JVM can load native code. This stipulation can cause a problem if, for example, you have an application (EAR file) with two web modules that both need to load the same native code through a Java Native Interface (JNI). Only the web module that first loads the library is successful.

To solve this problem, you can break out just the few lines of Java code that load the native code into a class on its own. Then, place this file on the WebSphere application class loader (in a utility JAR file). However, if you are going to deploy multiple such applications (EAR files) to the same application server (JVM), you must place the class file on the WebSphere extensions class loader instead. Doing so ensures that the native code is loaded only one time per JVM. The native code might be placed on a reloadable class loader (such as the application class loader or the WAR class loader). If so, it is important that the native code can properly unload itself should the Java code need to reload. WebSphere has no control over the native code and if it does not unload and load properly, the application might fail. If one native library depends on another one, things become even more complicated. Search for dependent native library in the information center for more details.

Class loader modes and policies

Class loader mode

- **PARENT_FIRST** (default): The class loader delegates the loading of classes to its parent class loader 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 is 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 2013

Figure 12-20. Class loader modes and policies

WA5913.0

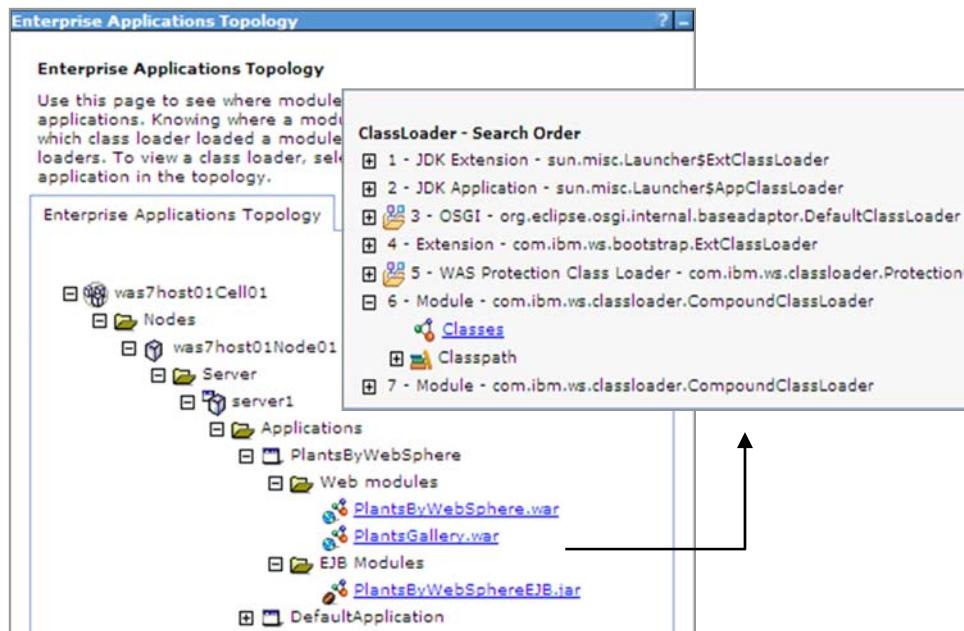
Notes:

The WebSphere Application Server V8.0 Information Center contains excellent information about class loading. To find more information, see:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp>



Class Loader Viewer: Application topology



© Copyright IBM Corporation 2013

Figure 12-21. Class Loader Viewer: Application topology

WA5913.0

Notes:

ClassLoader – Search Order: Use this page to examine the class loaders visible to a web module (.war file) or enterprise bean (.ejb file) in an installed enterprise application. This page helps you to determine which class loaders loaded files of a module and to diagnose problems with class loaders.

Enterprise Applications Topology: Use this page to see where modules are stored in a topology of enterprise applications. Knowing where a module is stored helps you to determine which class loader loaded a module and to diagnose problems with class loaders.

To view this administrative console page, click **Troubleshooting > Class loader viewer**. This page lists all installed applications and their modules in a tree view. The modules can be web modules (.war files) or enterprise bean (EJB) modules (.jar files).

When deploying an application to a server or starting an application, you might encounter problems that are related to class loaders. Use the console pages that are accessed from this page to troubleshoot errors such as:

- ClassCastException
- ClassNotFoundException

- NoClassDefFoundException
- UnsatisfiedLinkError

You can use the Class Loader Viewer console pages without having to restart or manipulate the application.

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 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 2013

Figure 12-22. Class loading problems (1 of 2)

WA5913.0

Notes:

An application can override classes that are contained in the parent class loader, but this action can potentially result in ClassCastException or LinkageErrors if you mix use of overridden classes and non-overridden classes.

See the information center topic *Class loading exceptions for details on how to troubleshoot these problems*: http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp?topic=%2Fcom.ibm.websphere.nd.multiplatform.doc%2Finfo%2Fae%2Fae%2Frtrb_classload_viewer.html

Class loading problems (2 of 2)

- A class loader problem is not 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 console, select **Troubleshooting > Class Loader Viewer**
- The Class Loader Viewer is structured as a hierarchy based on:
 - 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 that contains the class loader hierarchy
 - Each class loader that is shown contains a table view of the classpath it follows and classes it loads to satisfy the dependencies of the module

© Copyright IBM Corporation 2013

Figure 12-23. Class loading problems (2 of 2)

WA5913.0

Notes:

The hierarchy of class loaders that WebSphere Application Server V8.0 uses, from highest to lowest, is as follows:

1. Java class loaders
2. WebSphere extension class loaders
3. Application module class loader
4. Web module class loader

Three parameters control class loading of WebSphere Application Server V8.0 enterprise applications: 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 V8.0, *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.

Conversely, *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 use 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 is used to load classes for the web module. Changing the setting to *application* causes 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 use the same class loader, resulting in the sharing of resources globally among the applications.

The *isolation policy* has two 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.



How to resolve a ClassNotFound 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 ClassNotFound problem by enabling visibility to the class that cannot be found
- Trace the class loader by using the following trace string:

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

© Copyright IBM Corporation 2013

Figure 12-24. How to resolve a ClassNotFound problem

WA5913.0

Notes:

ClassNotFoundException: The class is not visible on the logical class path of the context class loader.

The class that is not found is not in the logical class path of the class loader that is associated with the current thread. The logical class path is the accumulation of all class paths searched when a load operation is invoked on a class loader. To correct this problem, use the Search page to search by class name and by Java archive (JAR) name:

1. Click **Troubleshooting > Class loader viewer > *module_name* > Search** to access the Search page.
2. For Search type, select **Class/Package**.
3. For Search terms, type the name of the class that is not found.
4. Click **OK**. The Search by class name page is displayed, listing all class loaders that load the class.
5. Examine the page to see whether the class exists in the list.

6. If the class is not in the list, return to the Search page. For Search terms, type the name of the .jar file for the class; for Search type, select **JAR/Directory**.
7. Click **OK**. The Search by Path page is displayed, listing all directories that hold the JAR file.
8. If the JAR file is not in the list, the class likely is not in the logical class path, is not readable, or another class is already loaded. Move the class to a location that enables it to be loaded.

Unit summary

Having completed this unit, you should be able to:

- Identify common problems with application installation
- Interpret runtime console messages and log messages
- Use application deployment troubleshooting tools
- Enable tracing of the appropriate components and interpret the data
- Use wsadmin to test and reproduce a problem
- Use the IBM Assembly and Deploy Tools for WebSphere Administration to validate application configuration files
- Use the Classloader Viewer tool to troubleshoot class loading issues
- Identify and troubleshoot class loader problems

© Copyright IBM Corporation 2013

Figure 12-25. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. What are some of the common problems that can occur when deploying an application?
2. What are the two parameters that control the class loading?
3. What are the three stages where problems might be encountered while deploying an application?

© Copyright IBM Corporation 2013

Figure 12-26. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. Some of the common problems are as follows:
 - Application is not visible.
 - Any validation error.
 - The application installation process times out.
 - A *NameNotFoundException* occurs. The newly deployed application does not respond to a request.
2. The following two parameters control the class loading:
 - Class loader mode
 - Class loader isolation policies
3. Problems might be encountered:
 - While installing, deploying, or updating the applications
 - When starting the application for the first time
 - When running the application for the first time

© Copyright IBM Corporation 2013

Figure 12-27. Checkpoint answers

WA5913.0

Notes:

Unit 13. Server start failures

What this unit is about

Application server start failure occurs because of system state, security, connectivity, or configuration problems. Such problems might be 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 success 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:

- Verify that an application server is failing to start
- Verify that a node agent is failing to start
- Verify that a deployment manager is failing to start
- Configure tracing on a server that fails to start
- Examine and analyze trace logs

How you will check your progress

- Checkpoint
- Lab exercises

References

WebSphere Application Server, Version 8.0 information center:

<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp>

IBM Troubleshooting Guide for WebSphere Application Server:

<http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg27005324>

Unit objectives

After completing this unit, you should be able to:

- Verify that an application server is failing to start
- Verify that a node agent is failing to start
- Verify that a deployment manager is failing to start
- Configure tracing on a server that fails to start
- Examine and analyze trace logs

© Copyright IBM Corporation 2013

Figure 13-1. Unit objectives

WA5913.0

Notes:



Topics

- Troubleshooting server start failures
- Troubleshooting server stop failures
- Node synchronization and node discovery failures
- Validate configuration

© Copyright IBM Corporation 2013

Figure 13-2. Topics

WA5913.0

Notes:

13.1.Troubleshooting server start failures

Troubleshooting server start failures



© Copyright IBM Corporation 2013

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

8.0

Figure 13-3. Troubleshooting server start failures

WA5913.0

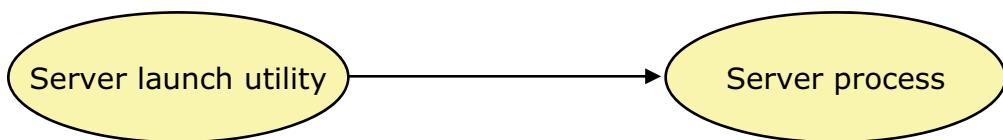
Notes:

After completing this topic, you should be able to:

- Verify that a server is failing to start
- Examine relevant logs for server start failure information
- Locate relevant server configuration files
- Use tools to validate server configuration files
- Identify and troubleshoot common server start failures
- Troubleshoot server stop failures

What happens when a server starts

- Issue the command: `startServer server1`
- Two JVMs are launched (except on iSeries)
 - Unless the `-script` parameter is specified
- The first JVM is the Systems Management server launch utility
 - Locates and reads the appropriate configuration (`server.xml`)
 - Creates a second JVM, which is the server process
- The server launch process waits until it receives back status from the server process, and then exits
 - Unless the `-nowait` parameter is specified



© Copyright IBM Corporation 2013

Figure 13-4. What happens when a server starts

WA5913.0

Notes:

For the OS/400 (i5/OS) operating system, there is *one* JVM. The OS/400 version of `startServer` calls a native program, which creates another native program that hosts the application server JVM. This configuration was done mainly for performance reasons, since two JVMs resulted in long startup times, not acceptable on OS/400. The native program runs some process setup and does the same function as the Systems Management server launch utility, collecting information that the application server needs. However, it does so natively, and then creates a JVM and calls the application server code from JNI.

Creating the process is achieved by constructing a command (`java -Dxxx -classpath x;y;z`) and running it.

z/OS notes:

- Server startup messages are in the job log (`SYSPRINT, SYSOUT`).
- There are start or stop server console commands on *z/OS*.
 - The XML files are in ASCII. If you plan to edit on UNIX System Services, convert the file to EBCDIC, edit, save, and convert back to ASCII by using the “`a2e, e2a`” tool.

- Any valid MVS user can start or stop the server from the console commands with security ON. No credentials are required on the command, unlike distributed.
- Stop, cancel, and force can be used to manage the server on z/OS.



Server start messages

The screenshot displays two separate Command Prompt windows, each showing the output of the 'startserver server1' command for a WebSphere Application Server profile named 'AppSrv01'. The top window shows a successful server launch, while the bottom window shows a failure due to a missing node agent.

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin>startserver server1
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\startServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 548
```

Server launch JVM is waiting for status back from the server


```
C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01\bin>startserver server1
ADMU0116I: Tool information is being logged in file C:\Program
Files\IBM\WebSphere\AppServer\profiles\AppSrv01\logs\server1\startServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3011E: Server launched but failed initialization. SystemOut.log<or job log in zOS> and other log files under C:\Program
Files\IBM\WebSphere\AppServer/profiles/AppSrv01/logs\server1 contain failure information.
```

- This particular failure was due to the node agent not running
- Exception is in SystemOut.log

© Copyright IBM Corporation 2013

Figure 13-5. Server start messages

WA5913.0

Notes:

In the successful server start shown here, 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 that is shown was the result of a node agent that was not started. This condition is clearly reported in the `SystemOut.log` and `SystemErr.log` files of server1.

Server bootup process

- A relatively small amount of WebSphere code is placed onto the class path of the JVM, including the bootstrap code
- The bootstrap code establishes:
 - A class loading environment for the rest of the installation image
 - In V6.1 and beyond, an OSGi environment is established so that the bundles are loaded
- The bootstrap code then branches into the “main” of the server
 - The “main” does some preliminary processing (like loading the `server.xml` document) and then begins starting the various components of the system

© Copyright IBM Corporation 2013

Figure 13-6. Server bootup process

WA5913.0

Notes:

The Open Service Gateway initiative (OSGi) defines an architecture for developing and deploying modular applications and libraries. OSGi is also known as Dynamic Module System for Java.

Both logging and FFDC are components and thus are not fully initialized until the bootstrap code branches into the main of the server. Failures during this early critical region can be challenging to troubleshoot.



Starting a server from the administrative console

- In a Network Deployment cell, application servers can 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 creates the server process
 - No data is streamed to the `startServer.log`



© Copyright IBM Corporation 2013

Figure 13-7. Starting a server from the administrative console

WA5913.0

Notes:

As of V7.0 there is a **Restart** link available in the administrative console.

Detecting a failed server start (1 of 3)

- An application server failure during the start process is reported as an error message in the command window or administrative console
- Error details are logged as exceptions in the server logs
 - native_stderr.log
 - native_stdout.log
 - startServer.log
 - SystemErr.log
 - SystemOut.log
- If the server started from the administrative console of the deployment manager, the errors might be viewable in the Runtime Error message panel
 - Select **Troubleshooting > Runtime Message > Error > Runtime Events**
 - Check the SystemOut and SystemErr log files of the node agent
- If the application server was started with the wsadmin script interpreter running, either the startServer.bat (Windows) or the startServer.sh (UNIX) script, the error message is printed in the command window

© Copyright IBM Corporation 2013

Figure 13-8. Detecting a failed server start (1 of 3)

WA5913.0

Notes:

Note on the <server_name>.pid file: You might assume that if this file is present, the server is definitely running. However, sometimes the file is left over after a server crash. Every application server, node agent, and deployment manager has a <server_name>.pid file in its logs directory when the server is running. This file is removed when the server is stopped gracefully by 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 continue to be fulfilled. Administrators might not detect a malfunctioning application server 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 a monitoring tool senses inactivity, an event is sent to the administrator in the form of an email or page.

One example of a Systems Management tool that can monitor the enterprise for failure is IBM Tivoli Composite Application Management, an enterprise-level monitoring and management application that can create historical reports about the performance of enterprise middleware. IBM Tivoli

Composite Application Management can send preconfigured events when thresholds are exceeded or rules are violated, and it does Systems Management tasks across a logical topology.

Detecting a failed server start (2 of 3)

- When starting an application server, various initialization steps occur
- The startup process is traced in the `SystemOut.log`
- 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

© Copyright IBM Corporation 2013

Figure 13-9. Detecting a failed server start (2 of 3)

WA5913.0

Notes:

During the start of an application server, various initialization steps occur. The startup process is traced in the `SystemOut.log`, 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.

Detecting a failed server start (3 of 3)

- Important message type prefixes:
 - WSVR (server runtime messages)
 - ADMU (server launch utility messages)

- Upon a successful application server start, you see the message:

```
ADMU3000I: Server <server name> open for e-business;
process id is <pid>
```

- It is possible that no errors are seen in the logs
 - An invalid JVM parameter might be the cause

© Copyright IBM Corporation 2013

Figure 13-10. Detecting a failed server start (3 of 3)

WA5913.0

Notes:

WSVR (server runtime messages) and ADMU (server utility messages) prefixes the message types of interest.

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 that is displayed depends on the cause of the failure.

It is important to point out that a server start failure, when done 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 results in no output files and the following console message:

```
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 message does not provide 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.

Starting a server from the console - failure

The screenshot shows the 'Application servers' page in the WebSphere Application Server console. In the 'Messages' section, a message states: 'was8host01Node01/MyAppServer server could not be started. View JVM logs for further details.' Below this, the 'Application servers' table lists two servers: 'MyAppServer' and 'server1'. The 'MyAppServer' row has a red error icon next to it. A yellow callout box points to the 'native_stderr.log' and 'native_stdout.log' files in the file list below the table, with the text: 'In this case, no JVM logs were created'.

Name	Size	Type	Date Modified
MyAppServer.pid	KB	PID File	2/29/2012 3:08 PM
native_stderr.log	KB	Text Document	2/29/2012 3:08 PM
native_stdout.log	KB	Text Document	2/29/2012 3:08 PM

© Copyright IBM Corporation 2013

Figure 13-11. Starting a server from the console – failure

WA5913.0

Notes:

This start failure was caused by setting the minimum and maximum heap of the server to 1 MB. 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.

Common causes of server start failures

Issue	Possible cause
Port conflict	<ul style="list-style-type: none"> Another process is using the port Server is already started and running Server is already started but JVM is hung orphaned
Node agent is not running	<ul style="list-style-type: none"> Application servers must register with the location service daemon on the node agent
Class loader problem	<ul style="list-style-type: none"> The class path is incorrect or classes are missing
Out of memory	<ul style="list-style-type: none"> Not enough native heap JVM maximum heap size is too small
Invalid JVM generic arguments or custom properties	<ul style="list-style-type: none"> JVM generic arguments are case-sensitive Non-IBM JDKs recognize different generic arguments
Server running as non-root	<ul style="list-style-type: none"> JVM does not have permissions to access required files (config, properties, logs, or other)
Application cannot start	<ul style="list-style-type: none"> Database incorrectly configured Transactions cannot be recovered

© Copyright IBM Corporation 2013

Figure 13-12. Common causes of server start failures

WA5913.0

Notes:

Before WebSphere Application Server V6.1 any server might fail to start because of an authentication failure, possibly due to an invalid server principal ID or because the LDAP server might not be reached. In V6.1 and later, exceptions might be thrown on server startup, but the server starts.

The JVM generic arguments are also referred to as JVM command-line options. See the *IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 6 Diagnostic Guide* for the details on using command-line options.

Determining server start issues (1 of 3)

- Begin with investigation of the log files to locate any exceptions
- Common server start failures will log exceptions
 - Usually, multiple exceptions occur during a server start failure
 - The initial failure is the general exception that accompanies a more detailed exception regarding the actual cause of the failure

© Copyright IBM Corporation 2013

Figure 13-13. Determining server start issues (1 of 3)

WA5913.0

Notes:

Various techniques can help you determine the root cause of an application server start failure. The log files are always the first place to start with a server start problem. Some server start problems are so severe that the Java virtual machine (JVM) fails 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 might be a combination of several problems, such as an error in configuration that causes a connectivity problem or security problem. For example, if the server does not stop properly, leaving it in a bad state, a connectivity problem might arise during restart because a necessary port is in use, which leads to the error message.

An example of a system state problem is reflected in the exception:

- `ADM10012E: Exception attempting to get free port for status socket {0}`

Explanation: An exception that occurs 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 is detected on port {0}
 - Likely causes:
 - An instance of the server {1} is already running
 - Some other process is using port {0}
 - Explanation: A port that this server requires is already in use by this server or another server.
 - User response: Do not attempt to use a port that is already in use.
- ADMU3027E: An instance of the server might already be running: {0}
 - Explanation: An attempt is 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 startup: com.ibm.ws.exception.RuntimeError: com.ibm.ws.exception.RuntimeError: com.ibm.ejs.EJS
 - Exception: Unable to register with location service daemon, which can only be in the node agent.
 - Make sure that the node agent for this node is running; nested exception is: org.omg.CORBA.ORBPackage.InvalidName
 - Name: LocationService:org.omg.CORBA.TRANSIENT:

```
java.net.ConnectException: Connection refused: connect:host=<host>,port=9900
vmcid: IBM minor code: E02 completed: No
```

Improperly specified or missing credentials is an example of a configuration problem that leads to a security problem, which can lead to the following error message:

- ADMU4113E: Verify that user name and password information is on the command line (-username and -password) or in the client.props file
 - Explanation: The tool was unable to contact the remote server because of a credentials problem.
 - User response: You might need to pass -username and -password on the command line, or update the soap/sas.client.props file.

However, a connectivity problem also might be a result of incorrect configuration. If one component is expecting to communicate on a port that is not correctly specified, the following error might occur:

- ADMU3040E: Timed out waiting for server initialization: {0} seconds
 - Explanation: The server initialization process exceeds 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 exceeds the timeout limit.
- User response: Perform problem determination on the server to see why it is taking so long to start.
- ADMIL0060W: Cannot contact server "{0}"
 - Explanation: The node agent failed to contact the server and then assumed that server was malfunctioning. The server is restarted if the "autoRestart" attribute is set to "true" in its monitoring policy, which is defined in the `server.xml` file.
 - User response: Check the `SystemOut.log` file for this server. Normally the log file shows why the server was not able to respond to the node agent. In case "autoRestart" is set to true, check to see whether the node agent restarted the server.

General configuration problems in which path, server, or any other configuration information is improperly specified might result in an error message such as:

- ADMU3402E: Server name is not specified; must supply the name of a server
 - Explanation: This tool requires a server name to 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 is defined for server: {0}
 - Explanation: An attempt was 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 the specified 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 might also occur. For simplicity, error messages from the ADMU family are explained in this unit.

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 streams 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 useful because some failures occur before initialization of the SystemOut.log and SystemErr.log trace streams
- If the server is started from the administrative console, there is no startServer.log
 - Instead, check the native_std*.log, and the node agent logs

© Copyright IBM Corporation 2013

Figure 13-14. Determining server start issues (2 of 3)

WA5913.0

Notes:

Specifying 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 that the server starts 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\rallanCell01\clusters//vari  
ables.xml not found  
java.io.FileNotFoundException:  
C:\WebSphere6\AppServer\profiles\AppSrv01\config\cells\rallanCell01\clusters\varia  
bles.xml (The system cannot find the path specified)  
at java.io.FileInputStream.open(Native Method)
```

The system was unable to locate a file during initialization time, leading to the initial failure message. The impact that this failure had on the overall system is not clear because it did start without this file.

Determining server start issues (3 of 3)

- Enable server verbose modes from the administrative console
 - Verbose class loading
 - Verbose garbage collection
 - **Application servers > *server_name* > Process Definition > Java Virtual Machine**
- Trace information is written to `native_stderr.log` by default
- If the administrative console is not available, edit the `server.xml` configuration file with these changes:
 - `verboseModeClass="true"`
 - `verboseModeGarbageCollection="true"`

© Copyright IBM Corporation 2013

Figure 13-15. Determining server start issues (3 of 3)

WA5913.0

Notes:

Verbose JNI can also be enabled, which provides debug output for native method invocation.

Common configuration error messages

Error	Message
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 <code>server.xml</code> for server
ADML0003E	A configuration error occurred in the <code>ProcessDef Umask</code> property
ADML0004E	An exception occurred when attempting to expand variable
ADML0006E	The <code> \${WAS_INSTALL_ROOT} </code> variable is missing

© Copyright IBM Corporation 2013

Figure 13-16. Common configuration error messages

WA5913.0

Notes:

Some common configuration error messages and their resolution are listed:

- Error: ADMU3402E: Server name not specified; must supply the name of a server
 Resolution: The command is ambiguous without a server name specified. This error is common when starting a server from `startServer.[bat|sh]` script in a deployment manager environment. Retrieve the server name from the `server.xml` file or the server name that is 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 spelling is not the problem, 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 in

the correct directory. If a symbolic link references the directory, 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 the application “run as” user and read the `server.xml` file?

- Error: ADML0003E: A configuration error occurred in the ProcessDef Umask property {0}

Resolution: The permissions under which the Java process runs is incorrect. Therefore, the Java process cannot access the 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 problems can be corrected in the administrative console by going to **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 specifies the actual variable in error. Go to the **Environment > WebSphere Variables** panel and correct the variable.

- Error: ADML0006E: The `${WAS_INSTALL_ROOT}` variable is missing

Resolution: Go to **Environment > WebSphere Variables** and define the `WAS_INSTALL_ROOT` variable.

Resolving server start failures (1 of 2)

Issue	Resolution
Port conflict	<ul style="list-style-type: none"> Verify server ports in <code>serverindex.xml</code> of the node Check for used ports with <code>netstat -a</code>, <code>lsof -i</code> or port scanning tools Use the <code>ps -ef</code> command on UNIX or the Task Manager on Windows to determine orphan processes and kill them
Node agent is not running	<ul style="list-style-type: none"> Start the node agent Check for orphaned node agent process, kill, and restart node agent
Class loader problem	<ul style="list-style-type: none"> Enable verbose class loading and look for <code>ClassNotFoundException</code> exceptions
Out of memory	<ul style="list-style-type: none"> Locate Java core dump and analyze with a thread analyzer tool Locate heap dump and analyze with a heap analyzer tool

© Copyright IBM Corporation 2013

Figure 13-17. Resolving server start failures (1 of 2)

WA5913.0

Notes:

A port scanning tool can be downloaded and started from the IBM Support Assistant.

Resolving server start failures (2 of 2)

Issue	Resolution
Invalid JVM generic arguments or custom properties	<ul style="list-style-type: none"> • Check the <code>server.xml</code> file for the application server for generic JVM arguments, remove them and start server • Modify from the administrative console and synchronize the node
Server running as non-root	<ul style="list-style-type: none"> • Fix the file permissions or • Add server user id to the appropriate group • Also, see the information about running as a service
Application cannot start	<ul style="list-style-type: none"> • Stop application from starting automatically on server start • Fix the cause of the application not starting

© Copyright IBM Corporation 2013

Figure 13-18. Resolving server start failures (2 of 2)

WA5913.0

Notes:

The JVM generic arguments are also referred to as JVM command-line options. See the *IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 6 Diagnostic Guide* for the details on using command-line options.

Determining why an application does not start is a separate topic. After the server starts, you can manually start the application. One problem might be timing that is related to the use of resources, including memory.

 WebSphere Education



IBM Port Scanning Tool

The results below show the port(s) that are not available for use because they are either active or configured.

PORt RANGE

Examples: 80 or 80-100 or 80, 90, 100 or you may leave [Advanced Options](#) the field blank to see all ports.

 [Print Results](#)

Port	Protocol	Status	Name	Process ID	Description	IP Version
8008	TCP	Active	httpd.exe	1476	C:\Program Files\IBM\HTTPServer\bin\httpd.exe	IPv4
8878	TCP	Active	java.exe	5656	C:\WebSphere\AppServer\java\bin\java.exe	IPv4
8879	TCP	Active	java.exe	4080	C:\WebSphere\AppServer\java\bin\java.exe	IPv4
8880	TCP	Active	java.exe	1744	C:\WebSphere\AppServer\java\bin\java.exe	IPv4
9043	TCP	Active	java.exe	4080	C:\WebSphere\AppServer\java\bin\java.exe	IPv4

© Copyright IBM Corporation 2013

Figure 13-19. IBM Port Scanning Tool

WA5913.0

Notes:

The IBM Port Scanning Tool is available from the IBM Support Assistant. It scans for active and some configured ports on the computer, providing relevant information about each port it encounters that is active or configured.

You can use the IBM Port Scanning Tool to scan for available ports during installation, configuration, or activation of an IBM product. You can also retrieve more information about a specific port on the computer.

13.2.Troubleshooting server stop failures

Troubleshooting server stop failures



© Copyright IBM Corporation 2013

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

8.0

Figure 13-20. Troubleshooting server stop failures

WA5913.0

Notes:

After completing this topic, you should be able to:

- Identify problems that might result in server stop failures
- Use different methods for stopping unresponsive or hung servers
- Restart an application server in recovery mode

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 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 might be hung and not respond to a stop command
 - Check `SystemOut.log` files for messages from the ThreadMonitor such as:

```
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
```

© Copyright IBM Corporation 2013

Figure 13-21. Server stop failures

WA5913.0

Notes:

In general, if a server fails to stop when appropriate user credentials are provided, the server is hung, and you should proceed by using the methods for troubleshooting a hung server.

WebSphere Education

Other administrative console stop server options

- **ImmediateStop**
 - Stops the server
 - Bypasses the normal server quiesce process that supports in-flight requests to complete before shutting down the entire server process
- **Terminate**
 - Ends the server process
 - Always attempt ImmediateStop before using this option

The screenshot shows the 'Application servers' page of the WebSphere Administrative Console. The toolbar at the top includes buttons for New..., Delete, Templates..., Start, Stop, Restart, ImmediateStop, and Terminate. The 'ImmediateStop' and 'Terminate' buttons are highlighted with a red box. Below the toolbar is a table with columns for Select, Name, Node, Host Name, Version, Cluster Name, and Status. Two rows are listed: 'MyAppServer' and 'server1'. Both rows show a checked checkbox in the 'Select' column. The 'Status' column for 'MyAppServer' has a red asterisk icon, while for 'server1' it has a green checkmark icon. At the bottom of the table, it says 'Total 2'.

© Copyright IBM Corporation 2013

Figure 13-22. Other administrative console stop server options

WA5913.0

Notes:

In addition to the **Stop** link, the administrative console provides **ImmediateStop** and **Terminate** options. **ImmediateStop** is faster than normal server stop processing, but application clients might receive exceptions. **Terminate** deletes the server process, and application clients might receive exceptions.

The **ImmediateStop** capability is also available when using wsadmin.

- Using JACL: \$AdminControl stopServer serverName immediate
- Using Jython: AdminControl.stopServer(serverName', immediate)'

Stop server by killing the Java process

- If an application server is hung, you can stop it by killing the Java process
 - Attempt to trigger a thread dump since the `javacore` file contains useful diagnostic data
 - Obtain 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 Task Manager to stop the task
 - In a UNIX environment, use the `kill -9` command to kill the Java process
- After killing the hung Java process, restart the server
- Stopping a hung process is just a workaround for the real problem
- Examine any heap dumps and `javacore` files

© Copyright IBM Corporation 2013

Figure 13-23. Stop server by killing the Java process

WA5913.0

Notes:

In the Windows Task Manager, the process ID is not shown by default.

However, you can add the process ID (PID) from the **View > Select Columns** option.

A `javacore` file is a snapshot of a running Java process. The IBM Java SDK produces a `javacore` file in response to specific operating system signals. `Javacore` files show the state of every thread in the Java process, and the monitor information that identifies Java synchronization locks.

Restarting an application server in recovery mode

- Background:
 - When an application server instance with active transactions in progress restarts after a failure, the transaction service can use recovery logs to complete the recovery process
- Problem:
 - A catastrophic failure occurs that leaves `InDoubt` transactions
- Resolution:
 - Issue the command: `startServer <server_name> -recovery` from the `<profile_root>/bin` directory of the server
 - The application server restarts in recovery mode, performs transactional recovery, and shuts down
 - Any resource locks that the application server held before the failure is released

© Copyright IBM Corporation 2013

Figure 13-24. Restarting an application server in recovery mode

WA5913.0

Notes:

Each transactional resource maintains recovery logs that are used to rerun any `InDoubt` transactions and return the overall system to a self-consistent state.

When you restart an application server in recovery mode:

- Transactional resources complete the actions in their recovery logs and then shut down. This action frees up any resource locks held by the application server before the failure.
- During the recovery period, only the subset of application server functions that are necessary for transactional recovery to proceed are available.
- The application server does not accept new work during the recovery process.
- The application server shuts down when recovery is complete.
- The recovery process begins as soon as all of the necessary subsystems within the application server are available. If the application server is not restarted in recovery mode, the application server can start accepting new work as soon as the server is ready, which might occur before the recovery work is completed.

- To prevent the assignment of new work to an application server that is going through its transaction recovery process, restart the application server in recovery mode.
- If you want to be able to restart an application server in recovery mode, do the following steps before a failure occurs, and then restart the application server to enable your configuration changes.

Procedure: If a catastrophic failure occurs that leaves InDoubt transactions, use the `startServer server_name -recovery` command from the command line. This command restarts the server in recovery mode. You must run the command from the `profile_root/bin` directory for the profile with which the server is associated.

Results: The application server restarts in recovery mode, does transactional recovery, and shuts down. Any resource locks that the application server held before the failure occurred are released.

Windows services: Monitoring server processes

Windows services

- Optional when defining profile for stand-alone node, managed node, and deployment manager
- `addNode` command has option to add node agent services

Use `WASService.exe` command to create and configure services

- User must belong to Administrator group and have advanced rights:
 - Act as part of the operating system
 - Log on as a service

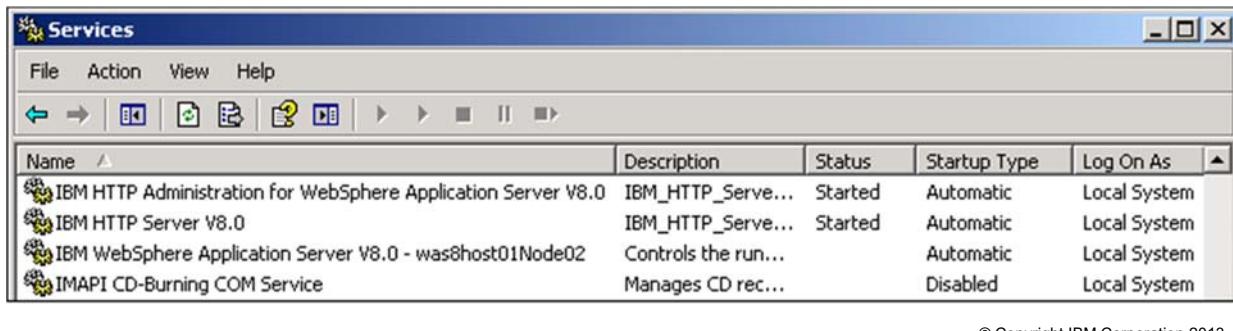


Figure 13-25. Windows services: Monitoring server processes

WA5913.0

Notes:

There are several server processes related to WebSphere Application Server products that the operating system can monitor and automatically restart when the server processes stop abnormally.

Using the installation wizard, you can create Windows services during installation. You can use the `WASService` command in the `install_root/bin` directory to create Windows services after installation. For more information, see the V8.0 information center.

You can configure a base Application Server as a WebSphere Application Server monitored process.

A tool to help create Windows services

- WASServiceHelper
 - A batch for creating, updating, and deleting WebSphere Application Server service on a Windows operating system
 - Use this tool to remove and re-create a Windows service for WebSphere
 - Helpful when a WebSphere Windows service does not start
- Run from the command line
- Available starting with WebSphere version 8.0
- Installed in <WAS_install_dir>\bin
 - The user is prompted for options
 - The `WASService.exe` command is displayed when complete

© Copyright IBM Corporation 2013

Figure 13-26. A tool to help create Windows services

WA5913.0

Notes:

There are several server processes related to WebSphere Application Server products that the operating system can monitor and automatically restart when the server processes stop abnormally.

Using the installation wizard, you can create Windows services during installation. You can use the `WASService` command in the `install_root/bin` directory to create Windows services after installation. For more information, see the V8.0 information center.

You can configure a base Application Server as a WebSphere Application Server monitored process.

UNIX: Automatic server monitoring and restart

- On a Linux or supported UNIX operating system, you can set up a shell script to automatically monitor and restart server processes
 - Locate the `rc.was` example shell script, which is in the `<was_root>/bin` directory
 - Create a shell script for each process that the operating system is to monitor and restart
 - Edit each shell script according to comments in its header, which provide instructions for identifying a product process
- Edit the `inittab` file of the operating system to add an entry for each shell script
 - Comments in the header of the `rc.was` file include a sample `inittab` entry line for adding this script to the `inittab` table
- Each `inittab` entry causes the operating system to call the specified shell script whenever the system initializes
- As each shell script runs, it monitors and starts the specified server

© Copyright IBM Corporation 2013

Figure 13-27. UNIX: Automatic server monitoring and restart

WA5913.0

Notes:

To automatically monitor and restart any related server processes on a Linux or supported UNIX operating system, set up a shell script.

- Locate the `rc.was` example shell script in the `app_server_root/bin` directory.
- Create a shell script for each process that the operating system is to monitor and restart.
- Edit each shell script according to comments in its header, which provide instructions for identifying a product process.
- Edit the `inittab` file of the operating system to add an entry for each shell script you created. Comments in the header of the `rc.was` file include a sample `inittab` entry line for adding this script to the `inittab` table. Each `inittab` entry causes the operating system to call the specified shell script whenever the system initializes. As each shell script runs, it monitors and starts the server process that you specified.

13.3.Node synchronization and node discovery failures

Node synchronization and node discovery failures



© Copyright IBM Corporation 2013

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

8.0

Figure 13-28. Node synchronization and node discovery failures

WA5913.0

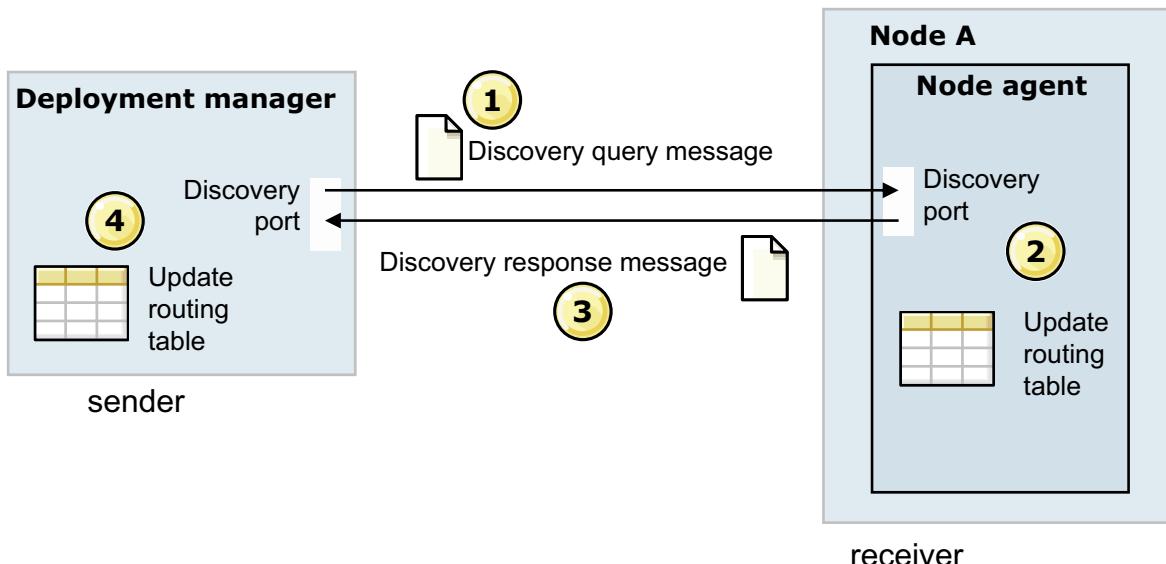
Notes:

After completing this topic, you should be able to:

- Identify problems that might result in server stop failures
- Use different methods for stopping unresponsive or hung servers
- Restart an application server in recovery mode

Node agent and deployment manager discovery process

- Each WebSphere Application Server process can be a “sender” or a “receiver” in the discovery process



© Copyright IBM Corporation 2013

Figure 13-29. Node agent and deployment manager discovery process

WA5913.0

Notes:

In this example, the deployment manager is the “sender” and the node agent is the “receiver”. Each message is broken down into multiple information components. Each message is represented in memory as ServerInfo and on the wire as XML.

1. The sender listens on the discovery port and sends a discovery query message to the peer discovery port. The retry interval is used if the peer does not respond.
2. The discovery query message is processed and the routing table is updated.
3. A discovery response message is sent to the sender’s discovery port.
4. The discovery response message is processed and the routing table is updated.

Important addresses in the discovery process

- **Cell_Discovery_Address**

- Used by the deployment manager to discover the node agent
- Defined in `serverindex.xml` of the deployment manager

```
<specialEndpoints  
xmi:id="NamedEndPoint_1" endPointName="CELL_DISCOVERY_ADDRESS">  
<endPoint xmi:id="EndPoint_1" host="websphere.ibm.com" port="7277"/>  
</specialEndpoints>
```

- **Node_Discovery_Address**

- Used by the node agent to discover the deployment manager
- Defined in the `serverindex.xml` of the node

```
<specialEndpoints xmi:id="NamedEndPoint_1089504933396"  
endPointName="NODE_DISCOVERY_ADDRESS">  
<endPoint xmi:id="EndPoint_1089504933407" host="websphere.ibm.com"  
port="7272"/>  
</specialEndpoints>
```

© Copyright IBM Corporation 2013

Figure 13-30. Important addresses in the discovery process

WA5913.0

Notes:

The discovery ports are entered in the `serverindex.xml` file of the deployment manager and node.

Failure of a managed node to synchronize

- Problem:

```
00002a04 NodeSync E ADMS0005E: The system is unable to generate
synchronization request:
javax.management.JMRuntimeException: ADMN0022E: Access is denied for the
getRepositoryEpoch operation on ConfigRepository MBean because of
insufficient or empty credentials.
```

- Cause:
 - Message usually indicates that LTPA keys were regenerated automatically
 - LTPA keys were not pushed correctly to the node
- “Quick fix” resolution:
 - Disable automatic generation of LTPA key
 - See the notes for complete resolution

© Copyright IBM Corporation 2013

Figure 13-31. Failure of a managed node to synchronize

WA5913.0

Notes:

To solve this problem, try disabling automatic generation of Lightweight Third Party Authentication keys. In the administrative console, disable “Automatically generate key” as follows:

1. **SSL certificate and key management > Key set groups > Select Key set group name.**
Clear the option to “Automatically generate keys”.
2. From the Key set groups, check **Key set group name**. Select the **Generated Keys** tab.
3. Click **OK** and **Save** to save the changes to the master configuration.
4. Stop the deployment manager.
5. From the `<profile_root>` of the deployment manager, delete the contents under the `wstemp`, `temp`, and `config/temp` folders.

6. Start the deployment manager.
7. From the `<profile_root>/bin` directory of AppServer, stop the node/server with `stopNode`/`stopServer` commands.
8. From `<profile_root>/bin`, manually synchronize the node by running `syncNode.sh`. Since security is enabled, run following command:

```
syncNode.sh <DMgr_hostName> <SOAP_PORT_of_DMGR> -username <username>
>Password <password>
```
9. Start the node and server.
10. Log on to the deployment manager administrative console, and verify the node and server availability.

As a preventive measure, set the automatic synchronization to occur more frequently. The resynchronization should occur within every two LTPA regeneration cycles. This process does not fix this problem if any nodes are offline for extended periods of time.

13.4. Validate configuration

Validate configuration



© Copyright IBM Corporation 2013

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

8.0

Figure 13-32. Validate configuration

WA5913.0

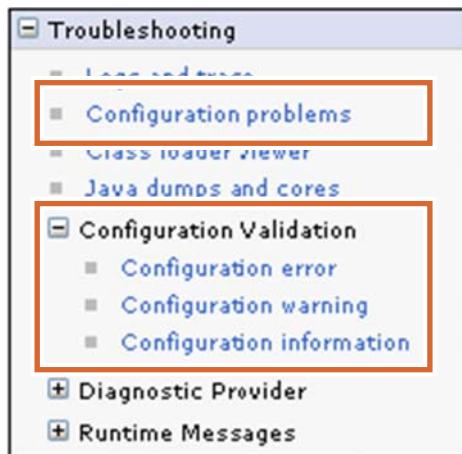
Notes:

After completing this topic, you should be able to:

- Validate the configuration of an application server through its configuration files

Monitoring configuration files from the administrative console

- There are two sections to monitor configuration problems from the administrative console
- Navigate to the **Troubleshooting** section



© Copyright IBM Corporation 2013

Figure 13-33. Monitoring configuration files from the administrative console

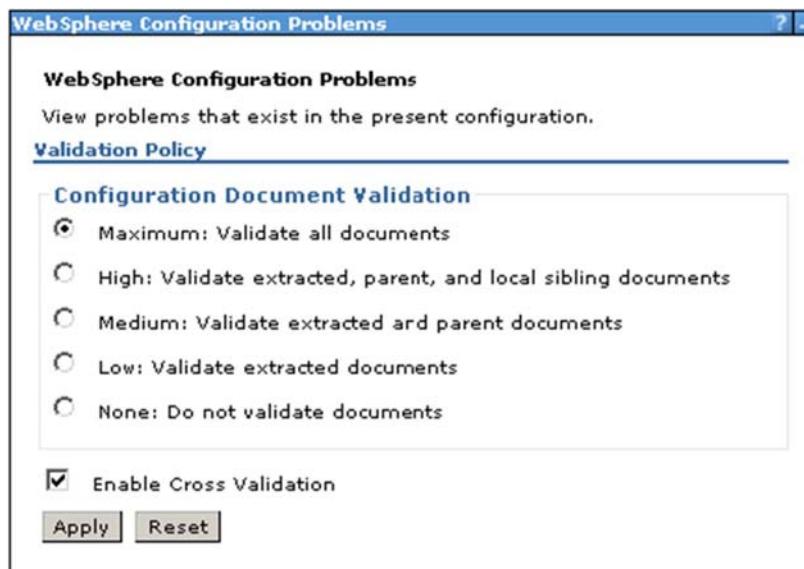
WA5913.0

Notes:



WebSphere Configuration Problems: validation policy

- Determine the level of validation to do on configuration documents
- Compare configuration documents for conflicting settings



© Copyright IBM Corporation 2013

Figure 13-34. WebSphere Configuration Problems: validation policy

WA5913.0

Notes:

The Validation Policy allows you to configure the validation level on configuration documents. The validation levels range from complete validation of all documents to no validation.

- **Maximum: Validate all documents**

This action turns on validation for all documents, regardless of whether they are extracted, and regardless of the relationships between the documents.

- **High: Validate extracted, parent, and local sibling documents**

This selection turns on validation for extracted documents and their parent documents, and turns on validation for the sibling documents of the documents that are extracted. For example, if **High** validation is selected, and if the `server.xml` document is extracted, when doing validation, validation is done on the three documents: `server.xml`, `node.xml`, and `cell.xml`. Validation is done also on the two sibling documents, `variables.xml` and `resources.xml`, within the `server1` directory.

- **Medium: Validate extracted and parent documents**

This action turns on validation for the documents that the user interface extracts, and also turns on validation of the parent documents of the documents that are extracted. For example, with

the partial directory structure from above, if **Medium** validation is selected, and if the `server.xml` document is extracted when validating, then validation is done on all three of the documents: `server.xml`, `node.xml`, and `cell.xml`.

- **Low: Validate extracted documents**

This action turns on validation for just those documents that the user interface extracts.

- **None**

Does not validate documents and disables validation. No configuration documents are validated.

The screenshot shows the 'WebSphere Configuration Problems' interface. On the left, there's a sidebar with a tree view:

- Troubleshooting**
 - Logs and trace
 - Configuration problems
 - Class loader viewer
 - Java dumps and cores
- Configuration Validation**
 - Configuration error** (highlighted with a red box)
 - Configuration warning
 - Configuration information
- Diagnostic Provider
- Runtime Messages

A large arrow points from the 'Configuration error' item in the sidebar to the error message listed in the main pane.

Main Pane: WebSphere Configuration Problems

View problems that exist in the present configuration.

+ Preferences

Configuration Problems

Filter: *server.xml*

You can administer the following resources:

CHKW3706E: Failed to validate contents of file cells/was8host01Cell01/nodes/ihsnod... Exception java.lang.NullPointerException	cells/was8host01Cell01/nodes/ ihsnod.../servers/webserver1/server.xml
--	--

Total 5 Filtered total: 1

© Copyright IBM Corporation 2013

Figure 13-35. Configuration problems

WA5913.0

Notes:



Unit summary

Having completed this unit, you should be able to:

- Verify that an application server is failing to start
- Verify that a node agent is failing to start
- Verify that a deployment manager is failing to start
- Configure tracing on a server that fails to start
- Examine and analyze trace logs

© Copyright IBM Corporation 2013

Figure 13-36. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. If the server start fails or partially fails, where might the error messages be streamed?
2. True or false: Server failure always results in output available in the `SystemErr.log` file.
3. Name the five common causes of server start failures.
4. What option do you specify as an argument to the `startServer` command to gather verbose information during the start process? Where does the information get logged to?

© Copyright IBM Corporation 2013

Figure 13-37. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

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

Checkpoint answers

1. The error messages could be streamed in any of the following places:
 - The `startServer.log`, `SystemErr.log`, or the `SystemOut.log` file
 - The administrative console
 - The command line console (shell)
2. False: Server failure does **not** always result in output available in the `SystemErr.log` file.
3. The following categories of issues might be present during server start failure:
 - Port conflict
 - Node agent is not running
 - Class loader issue
 - Out-of-memory
 - Invalid JVM generic arguments
4. The `-trace` option forces verbose information into the `startServer.log` file.

© Copyright IBM Corporation 2013

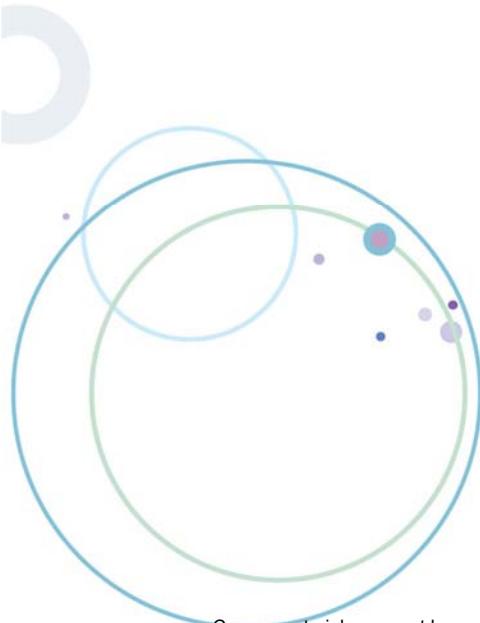
Figure 13-38. Checkpoint answers

WA5913.0

Notes:

Exercise 10

Troubleshooting server start failures



© Copyright IBM Corporation 2013

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

8.0

Figure 13-39. Exercise 10

WA5913.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Verify that an application server is failing to start
- Verify that a node agent is failing to start
- Verify that a deployment manager is failing to start
- Configure tracing on a server that fails to start
- Examine and analyze trace logs

© Copyright IBM Corporation 2013

Figure 13-40. Exercise objectives

WA5913.0

Notes:

Unit 14. Request flow and web container problems

What this unit is about

This unit follows a request through the system and explains various problems that can be encountered including the HTTP server, plug-in, and web container.

What you should be able to do

After completing this unit, you should be able to:

- Enable tracing as appropriate for various components in the end-to-end flow
- Describe pinging techniques for components in the request flow
- Enable HTTP plug-in tracing
- Describe how to bypass external HTTP servers
- Describe specific web container issues
- Use the WebContainerDP diagnostic provider

How you will check your progress

- Checkpoint
- Lab exercises

References

IBM Trace and Request Analyzer for WebSphere Application Server,
<http://www.alphaworks.ibm.com/tech/tra>

Support Authority: Real-time problem determination with WebSphere diagnostic providers,
http://www.ibm.com/developerworks/websphere/techjournal/0707_supauth/0707_supauth.html

Unit objectives

After completing this unit, you should be able to:

- Enable tracing as appropriate for various components in the end-to-end flow
- Describe pinging techniques for components in the request flow
- Enable HTTP plug-in tracing
- Describe how to bypass external HTTP servers
- Describe specific web container issues
- Use the WebContainerDP diagnostic provider

© Copyright IBM Corporation 2013

Figure 14-1. Unit objectives

WA5913.0

Notes:



Topics

- Overview of web request flow
- Troubleshooting a failing request
- Troubleshooting HTTP session issues
- Troubleshooting dynamic cache issues
- Using diagnostic providers to troubleshoot web container issues

© Copyright IBM Corporation 2013

Figure 14-2. Topics

WA5913.0

Notes:

14.1.Overview of web request flow

Overview of web request flow



© Copyright IBM Corporation 2013

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

8.0

Figure 14-3. Overview of web request flow

WA5913.0

Notes:

After completing this topic, you should be able to:

- Understand the web request flow
- Identify various ways of serving a request
- Interpret web server and plug-in log files
- Enable tracing of the web server plug-in and web container



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 2013

Figure 14-4. Web request protocol

WA5913.0

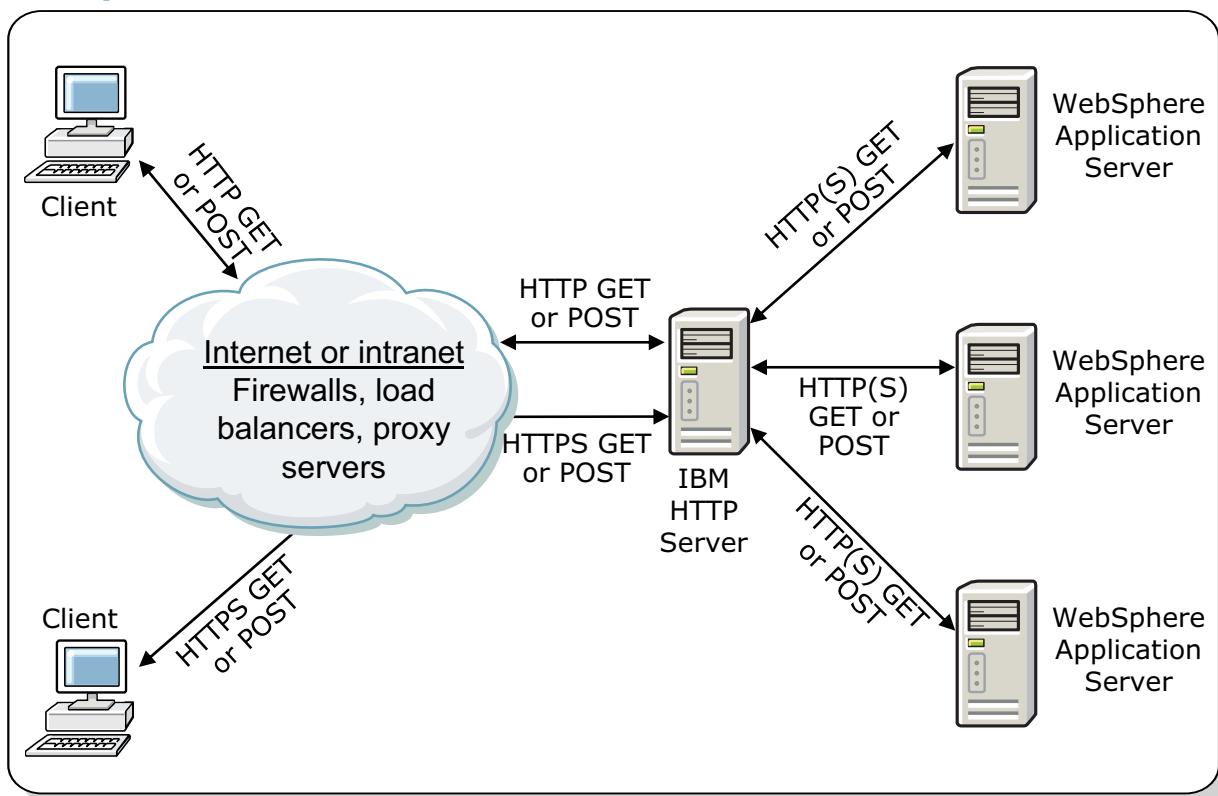
Notes:

There are three areas of interest:

- **Protocols** – This course is limited to the following two protocols:
 - HTTP is unsecured protocol. The default port for this protocol is 80.
 - HTTPS is secured protocol. The default port for this protocol is 443.
- **Methods** – This module focuses on GET and POST. Other methods are HEAD, PUT, and DELETE.
 - The GET request contains all request information in request headers. Request parameters are encapsulated in the URL. **No** further body is attached to the GET request.
 - The POST request has a POST body that is followed in addition to the request headers. The content-length header defines the length of POST body. All request parameters are present in body of the POST request.
- **Return codes** – These codes represent the common response codes that you notice in logs when debugging problems:
 - 100 – Continued response.

- 200 – Successful completion.
- 302 – Temporary redirection. Response contains a **Location** header with the redirected location.
- 304 – Not modified.
- 400 – Bad request. Client did not formulate 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.

Request flow HTTP or HTTPS



© Copyright IBM Corporation 2013

Figure 14-5. Request flow HTTP or HTTPS

WA5913.0

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 might also cause request failures. These concerns are not explored in this unit. If you suspect that one of these gateways is the cause of a request failure, you can prove or disprove this possibility. You can take a packet trace from the IBM HTTP Server box to verify the content and movement of a request. The WebSphere Application Server plug-in is supported for web servers like Microsoft IIS, Sun One WebServer, iPlanet, and others.

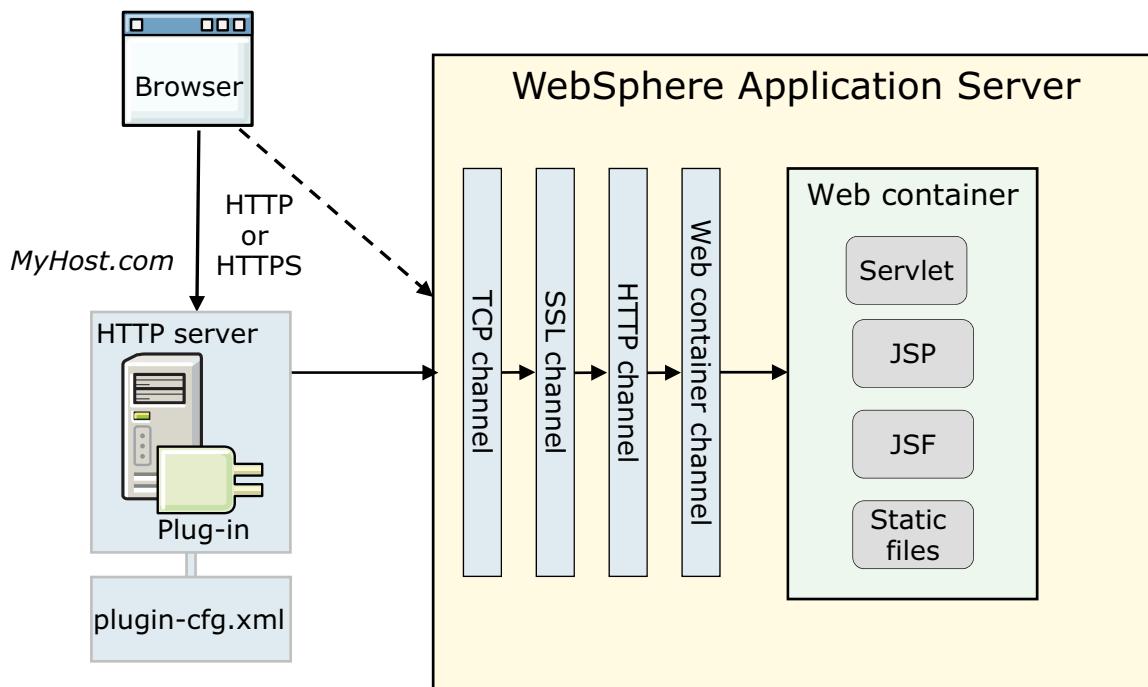
Packet traces are also known as IP traces. These traces record the individual packets from the TCP layer as they move in and out of the problematic box. They are most helpful when you are suspicious that requests are not making it to a problematic box, or that the content of the request does not represent what is expected. Reading a packet trace does require an experienced network administrator to interpret the data.

Other tips to find 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 the ping to the host, you might want to try Telnet to the 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 that the application server is listening on 9080 for HTTP traffic, here is a URL: `http://hostname:9080/myapplication.jsp`

Detailed web request flow



© Copyright IBM Corporation 2013

Figure 14-6. Detailed web request flow

WA5913.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server showing WebServer and key components in WebSphere Application Server that handle requests.

The key components in WebSphere Application Server are Channel framework and the web container. Channel framework replaces the HTTP Transport component in earlier versions (before V6).

Channel framework consists of various channels that interact in handling the socket data. A request goes through TCP, SSL (only for HTTPS), HTTP, and web container channels before the web container handles it.

- The TCP channel manages client connections, thus providing an 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 an 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 for WebSphere to handle, 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 and steer traffic away from unavailable servers. It reads the application server's configuration from a special XML file, `plugin-cfg.xml`.



Typical request methods: GET

- GET /PlantsByWebSphere/

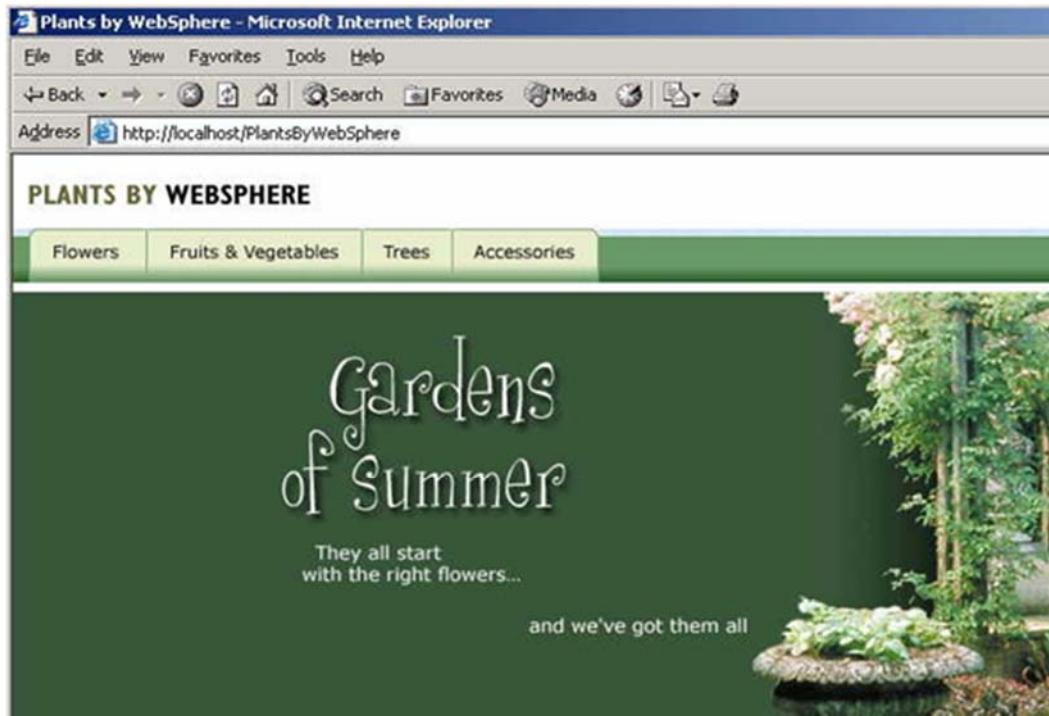


Figure 14-7. Typical request methods: GET

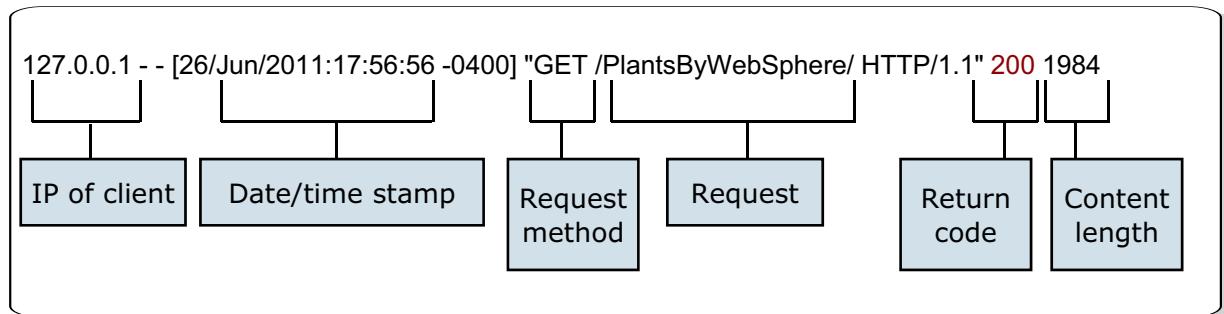
WA5913.0

Notes:

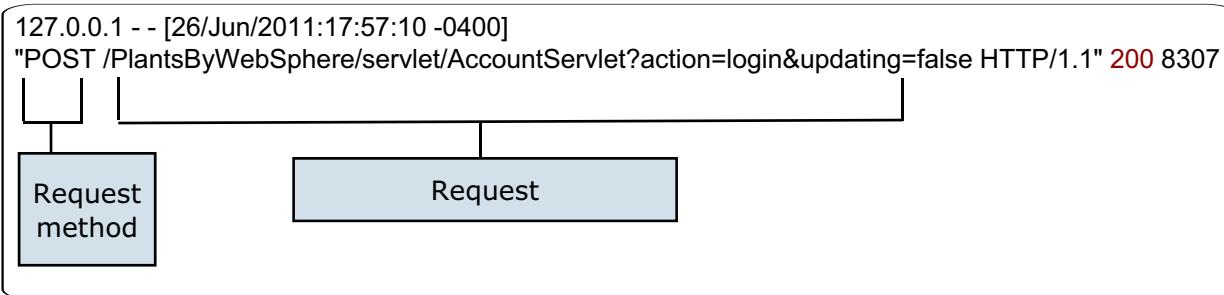
The screen shows the successful results for a GET request to the PlantsByWebSphere application. A GET is any request that seeks content from a host application.

Identifying a working request: Web server access log

- IBM HTTP Server access log for a successful `GET` request



- IBM HTTP Server access log for a successful `POST` request



© Copyright IBM Corporation 2013

Figure 14-8. Identifying a working request: Web server access log

WA5913.0

Notes:

All requests that are submitted to the IBM HTTP Server are recorded within the web server's access log at the completion of the request. Here 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 might not get posted. And obviously, if a request never makes it to the web server, it too does not get recorded.

If you want to track request information, you can change the LogLevel to debug in the configuration file. This tracking enables IBM HTTP Server to write detailed information in error.log.

The "-0400" within the access log represents this system's time offset from GMT time.

Return codes of 4xx and 5xx are failure codes.

IBM HTTP Server 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`.

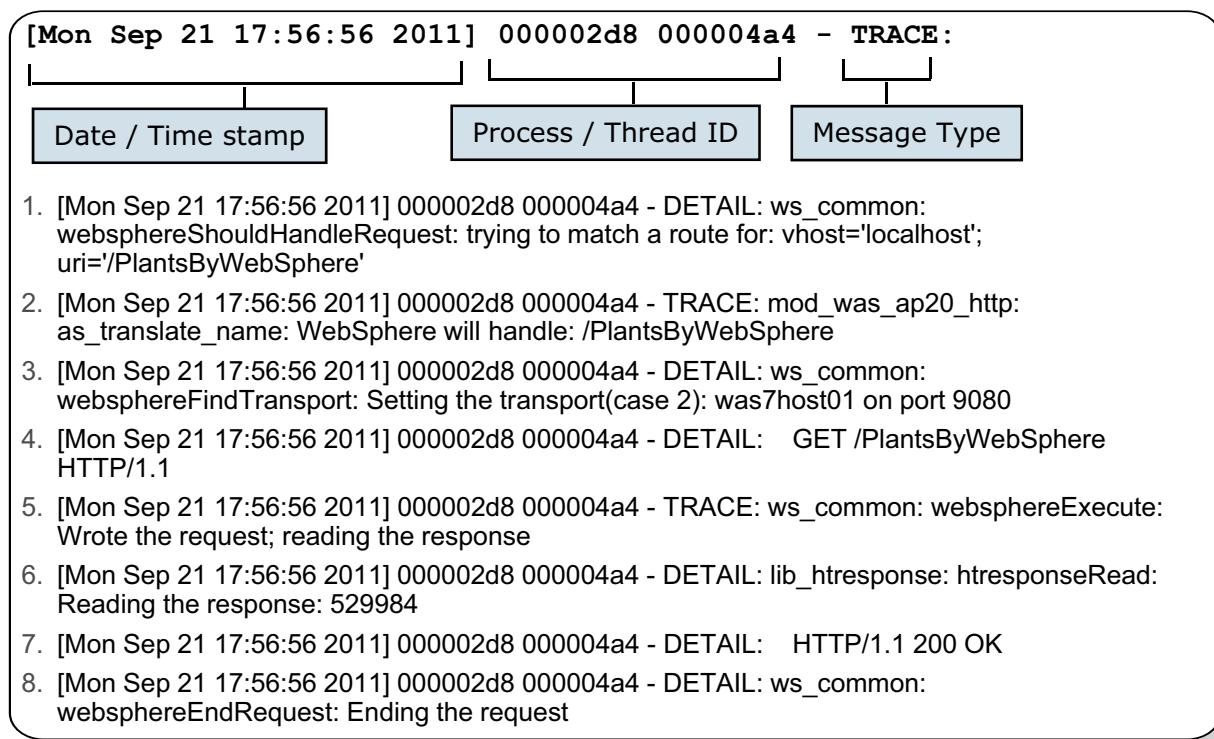
Both of the examples above show that a 200 response was returned. The output conforms to the default IBM HTTP Server configuration log format.

Notice that the file name for the access log does not provide the fully qualified path to the file. IBM HTTP Server assumes 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 that is associated with the *CustomLog* directive.

Another commonly used *LogFormat* for access log is “combined”. See the default `httpd.conf` file for details.

Identifying a working request: Plug-in log (GET)



© Copyright IBM Corporation 2013

Figure 14-9. Identifying a working request: Plug-in log (GET)

WA5913.0

Notes:

The lines in the image are various log statements from a plug-in log with LogLevel set to trace. Plug-in trace is not enabled by default; in the next few slides you 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 to validate that the plug-in received and processed a request.

The breakdown of each line is as follows:

1. Shows that the request was received.
2. The plug-in successfully matched the request against its criteria that are outlined within the `Plugin-cfg.xml` file.
3. The plug-in chose a transport host name of `App_Server_01` on Port 9080.
4. Shows the type of method that is submitted to the application server.
5. The request was submitted to the application server.
6. The plug-in is now reading the response from the application server.
7. The application server posted a response code to the plug-in.

8. The request is completed.

To get this level of detail, you must enable tracing within the `Plugin-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.

Identifying a working request: Plug-in log (POST)

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - DETAIL:

```

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - DETAIL:
|-----+-----+-----+
| Date / Time stamp | Process / Thread ID | Message Type |
|-----+-----+-----|

```

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - DETAIL: ws_common:
websphereShouldHandleRequest: trying to match a route for: vhost='localhost';
uri='/PlantsByWebSphere/servlet/AccountServlet'

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - TRACE: mod_was_ap20_http:
as_translate_name: WebSphere will handle: /PlantsByWebSphere/servlet/AccountServlet

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - DETAIL: ws_common: websphereFindTransport:
Setting the transport(case 2): was7host01 on port 9080

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - DETAIL: POST
/PlantsByWebSphere/servlet/AccountServlet?action=login&updating=false HTTP/1.1

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - TRACE: ws_common: websphereExecute: Wrote
the request; reading the response

[Wed Jun 27 16:32:12 2011] 0000047c 00000874 - DETAIL: lib_htresponse: htresponseRead:
Reading the response: 52997c

[Wed Jun 27 16:32:30 2011] 0000047c 00000874 - DETAIL: HTTP/1.1 200 OK

[Wed Jun 27 16:32:30 2011] 0000047c 00000874 - DETAIL: ws_common: websphereEndRequest:
Ending the request

© Copyright IBM Corporation 2013

Figure 14-10. Identifying a working request: Plug-in log (POST)

WA5913.0

Notes:

These lines are almost identical to the previous GET slide. The only difference is that the POST method was used.



Enabling IBM HTTP Server trace

General Properties

Web server name	webserver02
Type	IBM HTTP Server
Host name	krishna
* Port	80
* Web server installation location	C:/IBM/HTTPServer7
* Configuration file name	<code> \${WEB_INSTALL_ROOT}/conf/httpd.conf</code>
	<input type="button" value="Edit"/>

Enabling IBM HTTP Server trace

1. Open the administrative console
2. Go to **Servers > Server Types > Web servers > Server_Name > Edit configuration**
3. Change log level from **warn** to **debug**; save changes
4. Restart IBM HTTP Server
5. Trace information is written to **error.log**

© Copyright IBM Corporation 2013

Figure 14-11. Enabling IBM HTTP Server trace

WA5913.0

Notes:

It is not possible to change the log level for IBM HTTP Server dynamically. The IBM HTTP Server instance must be restarted to pick up this change. Here is an extract from httpd.conf file for the LogLevel directive:

```
#  
# LogLevel: Control the number of messages logged to the error.log.  
# Possible values include: debug, info, notice, warn, error, crit,  
# alert, emerg.  
#
```

If you do not use the administrative console to manage the IBM HTTP Server configuration, then you can edit the `httpd.conf` file manually and restart the web server.



Enabling plug-in trace

Plug-in logging:

* Log file name
C:\IBM\HTTPServer7\Plugins\logs\webserver02\http_plugin.log

Log level

Error

Trace

Stats

Warn

Error

Debug

Detail

Reset Cancel

Enabling plug-in trace

1. Open the administrative console
2. Go to **Servers > Server Types > Web servers > Server_Name > Plug-in properties**
3. Change log level from **Error** to **Trace**
4. Save changes
5. Regenerate and Propagate the plug-in configuration
6. Restart IBM HTTP Server
7. Trace information is written to **http-plugin.log**

© Copyright IBM Corporation 2013

Figure 14-12. Enabling plug-in trace

WA5913.0

Notes:

Plug-in trace can be enabled dynamically. By default, the refresh interval for the `plugin-cfg.xml` file is 60 seconds, so changes should get picked up dynamically. If changes are not picked up dynamically, you might be required to restart the web server.

If you do not use the administrative console to manage the IBM HTTP Server configuration, then you can edit the `plugin-cfg.xml` file manually.

Here is an extract from `plugin-cfg.xml` for LogLevel:

```
<Log LogLevel="Trace" Name="c:\IBM\WebSphere61\logs\http_plugin.log"/>
```

Available LogLevels are error, warning, stats, detail, debug, and trace.



Enabling web container trace

General Properties

Change Log Detail Levels

Components

Groups

```
com.ibm.ws.webcontainer=all:  
com.ibm.ws.webcontainer.*=all:  
com.ibm.wsspi.webcontainer=all:  
com.ibm.wsspi.webcontainer.*=all:  
HTTPChannel=all:GenericBNF=all
```

Enabling web container trace

1. Open the administrative console
2. Go to **Troubleshooting > Logs and Trace > Server_Name > Change Log Detail Levels**
3. In the Configuration tab, enter these trace strings:

```
com.ibm.ws.webcontainer=all:  
com.ibm.ws.webcontainer.*=all:  
com.ibm.wsspi.webcontainer=all:  
com.ibm.wsspi.webcontainer.*=all:  
HTTPChannel=all:GenericBNF=all
```

4. Save changes
5. Trace information is written to **trace.log**

© Copyright IBM Corporation 2013

Figure 14-13. Enabling web container trace

WA5913.0

Notes:

1. You can enable trace for various components in application server. The trace string that is shown enables traces for components that are of interest in handling the HTTP request. For detailed instructions on how to gather trace, you can refer to the website:
<http://www.ibm.com/support/docview.wss?rs=180&uid=swg21384592>
2. Trace can also be enabled dynamically. For detailed instructions on how to gather trace in various ways, you can refer to the website:
<http://www.ibm.com/support/docview.wss?rs=180&uid=swg21254706>



Web container diagnostic provider

- The 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



- 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 2013

Figure 14-14. Web container diagnostic provider

WA5913.0

Notes:

The IBM Education Assistant contains a module on the IBM WebSphere Application Server V6.1 Diagnostic Providers that is highly informative and contains further resources about the diagnostic providers. This educational module is available as part of the WebSphere Application Server V6.1 Information Center at:

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.was_v6/was/6.1/ProblemDetermination/WASv61_DiagnosticProviders/player.html

14.2.Troubleshooting a failing request

Troubleshooting a failing request



© Copyright IBM Corporation 2013

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

8.0

Figure 14-15. Troubleshooting a failing request

WA5913.0

Notes:

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

Preliminary questions (1 of 2)

1. What is the error code in the browser?
 - Possible error codes are DNS Error, 4xx, 5xx, or error message from application with success response code
2. Are all components running?
 - Verify that the web server and the WebSphere application are running
3. DNS is working correctly?
 - Verify that the web server host name and IP address are correct within DNS
4. What was the result of by-passing the web server?
 - If the problem does not happen when bypassing 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 lie between the client and the web server
 - One of these components could keep 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 might have an OS-related patch that is required to correct a problem

© Copyright IBM Corporation 2013

Figure 14-16. Preliminary questions (1 of 2)

WA5913.0

Notes:

Before you investigate a failing request, you need to research some important questions. These questions are listed and explained in the slide.

Question 1: What is the error code in the browser? This question is the key to further troubleshooting the problem. The response to this question helps to narrow down the rest of the questions.

Question 2: Are all components running? This question might seem foolish, but sometimes it happens that WebSphere Application Server is running, but the application is not.

Question 3: Is the DNS working correctly? Make sure that the host name of the request does map to the IP of the web server.

Question 4: What is the result of bypassing the web server? Identify the likely location of the problem source. Is it in the web server or the application server?

Question 5: Are there any firewall or front-end issues? Firewalls must 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: What are the operating systems of all components involved? When researching problematic requests, sometimes it becomes apparent that the correct maintenance updates for the operating system were not applied.

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 enter in the client browser?
 - Must know the exact request that was typed into the client browser
9. What browser are you using?
10. What results were returned?
 - Obtain a print screen of the results
 - Problems can result with the wrong response or no response
11. What was the date and time of the request?
 - This information helps to locate the problematic request within the logs
12. Who was supposed to respond to the request?
 - Web server or application server

© Copyright IBM Corporation 2013

Figure 14-17. Preliminary questions (2 of 2)

WA5913.0

Notes:

Question 7: What are the versions and maintenance levels of the involved products? Like question 6, problematic requests might be documented and resolved in later releases of your web server, plug-in, or application server updates.

Question 8: What did you type in? Knowing the exact request that is typed into the client browser is critical to diagnosing the problem. It also determines whether the request was SSL or not.

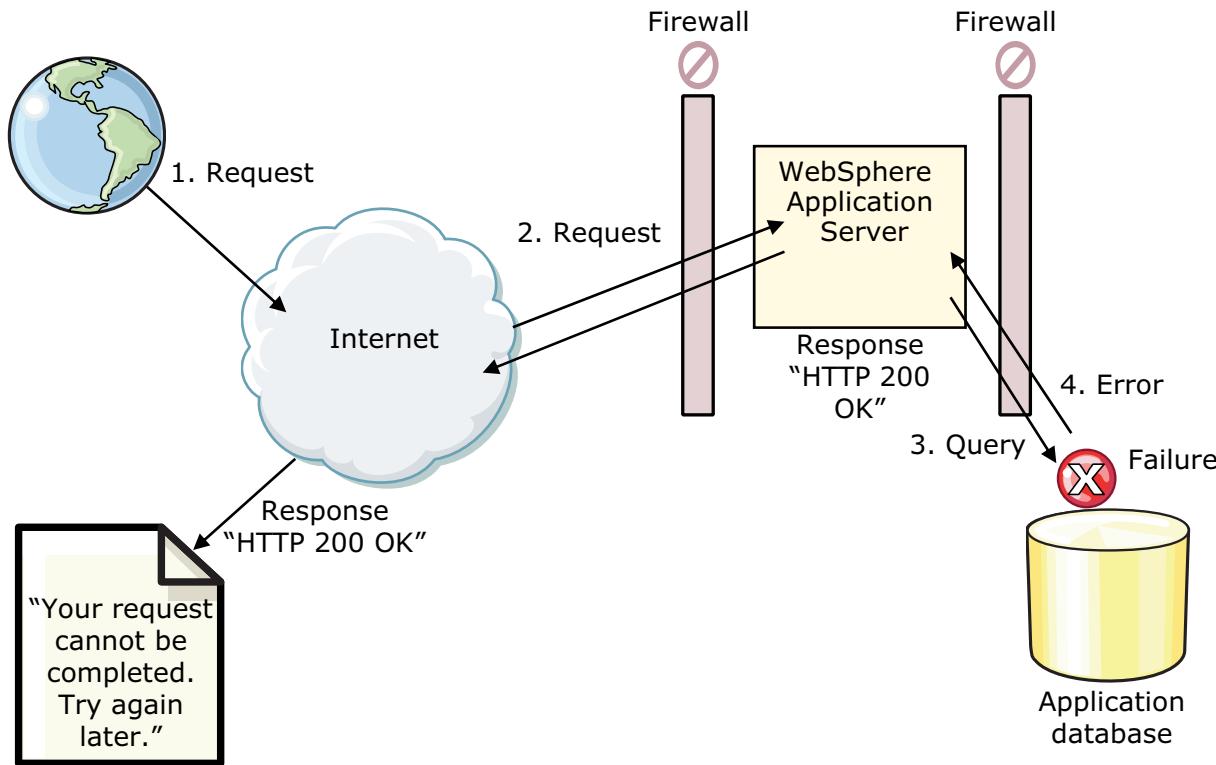
Question 9: What browser are you using? You need to have verification of the results. You are going to use this information to determine what type of return code was likely issued.

Question 10: What did you get back? This item is key for reviewing log files.

Question 11: What was the date and time of the request? This question is key to identifying what logs are necessary to troubleshoot the problem.

Question 12: Who was supposed to respond to the request, the web server or application server? Knowing your application architecture can expedite where to search first.

A word about HTTP response codes



© Copyright IBM Corporation 2013

Figure 14-18. A word about HTTP response codes

WA5913.0

Notes:

The HTTP response codes might not be helpful in tracking down the problem because all response messages might be "200 OK". It is common for correctly designed enterprise websites to handle errors at the web server or application server. An easy-to-use web page is then returned to the customer, explaining the nature of the problem. As a result, website monitoring tools typically provide configuration parameters that take strings as input, and trigger an event that is based on finding that string in the web page. For example, a database server might be unavailable, which causes the transaction to fail. As a result, the monitoring tool looks through the returned web page for "Unable to complete your request" or some error message meant to inform the user that a problem occurred.

Web server generates a 404 response

- IBM HTTP Server example for the following request:
 - `http://localhost/mypage.html`

<u>Response header</u>
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 `<IHS Root>/logs/access.log`
 - `127.0.0.1 - - [21/Sep/2009:17:49:59 -0400] "GET /mypage.html HTTP/1.1" 404 280`
- Locate the error within `<IHS Root>/logs/error.log`
 - `[Mon Sep 21 17:49:59 2009] [error] [client 127.0.0.1] File does not exist: C:/ProgramFiles/IBM/HTTPServer/htdocs/en_US/mypage.html`

© Copyright IBM Corporation 2013

Figure 14-19. Web server generates a 404 response

WA5913.0

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 suggested that you always start reviewing the web server logs first when debugging nonworking requests. Requests served only by the web server are classified as static requests and usually consist of HTML pages and image files. Requests that WebSphere serves are classified as dynamic requests, but can also include static content. For this unit, if the web server serves content, it is referred to as static; and if the application server serves content, it is referred to as dynamic.

The following are key things to know ahead of time:

1. What file was requested (for example, `mypage.html`)
2. Date and time the problem occurred
3. The IP address of the client browser

Web server generates a 404 response: Solution

- Verify the documentRoot path in the IBM HTTP Server's httpd.conf configuration file (*<IHS root>/conf/httpd.conf*) 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 is 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 locale might qualify the path to the resource
 - A resource found in the documentRoot cannot be located if the URL is rewritten to include the locale
 - For example, a request for somedir/mypage.html should be in the directory somedir/en_US/mypage.html after the URL localization occurs for a user agent that is requesting the US_en version of the page

© Copyright IBM Corporation 2013

Figure 14-20. Web server generates a 404 response: Solution

WA5913.0

Notes:

If the static content you are requesting does not exist under the *DocumentRoot*, then a 404 Not Found error occurs. You must make sure that your content exists under the *DocumentRoot* to avoid these types of errors. You can 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



Plug-in fails to route a request to application server

- The IBM HTTP Server generates a “404 Not Found” error when the request should have been sent to the server:
 - `http://localhost/PlantsByWebSphere`
- Locate the request within `<IHS Root>/logs/access.log`:
 - `127.0.0.1 - - [21/Sep/2009:17:59:01 -0400]`
`"GET /PlantsByWebSphere HTTP/1.1" 404 305`
- Search for “File does not exist” errors in the `<IHS Root>/logs/error.log`:
 - `[Mon Sep 21 17:59:01 2009] [error] [client 127.0.0.1]`
`File does not exist: C:/Program Files/IBM HTTP Server/`
`htdocs/en_US/PlantsByWebSphere`

© Copyright IBM Corporation 2013

Figure 14-21. Plug-in fails to route a request to application server

WA5913.0

Notes:

This example shows that the web server attempted to serve a WebSphere request from its document root.

A “file does not exist” error 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 then tries to serve the request from its *DocumentRoot* where it obviously does not exist, usually because of an incorrect configuration of the `plugin-cfg.xml` file. See the next slide.

Plug-in causing a 404 response: HTTP plug-in

- Locate the request in the `http_plugin.log` file to determine why the HTTP plug-in failed to send the request to WebSphere:

```

- [Mon Sep 21 17:59:01 2009] 000002d8 000004a8 - DETAIL:
  ws_common: websphereShouldHandleRequest: trying to match a
  route for: vhost='localhost'; uri='/PlantsByWebSphere'

- [Mon Sep 21 17:59:01 2009] 000002d8 000004a8 - TRACE:
  ws_common: websphereVhostMatch: Comparing '*:80' to
  'localhost:80' in VhostGroup: default_host

- [Mon Sep 21 17:59:01 2009] 000002d8 000004a8 - DEBUG:
  ws_common: websphereVhostMatch: Found a match '*:80' to
  'localhost:80' in VhostGroup: default_host with score 1,
  exact match 0

- [Mon Sep 21 17:59:01 2009] 000002d8 000004a8 - TRACE:
  ws_common: websphereUriMatch: Failed to match:
  /PlantsByWebSphere

- [Mon Sep 21 17:59:01 2009] 000002d8 000004a8 - DETAIL:
  ws_common: websphereShouldHandleRequest: No route found

```

© Copyright IBM Corporation 2013

Figure 14-22. Plug-in causing a 404 response: HTTP plug-in

WA5913.0

Notes:

This 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_URLs`, 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. The web server tries to serve the request from its document root, as seen in the previous slide. The result is a 404 error that the web server generates.

An application server might return a 404 error message 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 was mapped to a wrong virtual host or invalid alias in mapped virtual host.
- `Plugin-cfg.xml` contains wrong information and routes request to wrong server.
- Use `WebContainerDP` to view configuration data of the web container. It should contain all information about virtual host mappings.

- For all cases where the application server returns 404, you see information similar to this information in the plug-in trace:

```
[Mon Sep 21 17:57:48 2009] 000002d8 000004a8 - DEBUG: lib_htrequest:  
htrequestWrite: Writing the request:  
[Mon Sep 21 17:57:48 2009] 000002d8 000004a8 - DETAIL: GET  
/PlantsByWebSphere/GetPlant.jsp HTTP/1.1  
[Mon Sep 21 17:57:48 2009] 000002d8 000004a8 - DETAIL: Accept: image/gif,  
image/x-xbitmap, image/jpeg, image/pjpeg, */*  
.... . . . . .  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: HTTP/1.1 404 Not Found  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: Content-Type:  
text/html; charset=ISO-8859-1  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: $WSEP:  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: Content-Language: en-US  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: Transfer-Encoding:  
chunked  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: Connection: Close  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: Date: Tue, 26 Jun 2009  
21:57:51 GMT  
[Mon Sep 21 17:57:51 2009] 000002d8 000004a8 - DETAIL: Server: WebSphere  
Application Server/7.0
```

Application server generates a 404 response

- Problem: Requested (or dependent) resource not found
- Symptoms:
 - 404 return code in `access.log`
 - Plug-in trace shows application server that sent 404 response
 - `SystemOut.log` might not contain any information
 - Application server trace shows similar information as:

```
[6/19/09 21:50:38:274 EDT] 0000001d WebApp 3 Exception
  type=java.io.FileNotFoundException
[6/19/09 21:50:38:274 EDT] 0000001d WebApp 3 Exception message=SRVE0190E:
  File not found: /dummyTest
```
- Possible solutions include verifying:
 - The validity of the URL
 - The context root of web application
 - The file exists in the web application
 - The resource is defined in `web.xml`
 - The correct version of the `plug-in.xml` file is referenced and not an older version

© Copyright IBM Corporation 2013

Figure 14-23. Application server generates a 404 response

WA5913.0

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 via WebServer plug-in, `fileServing` feature should be enabled. So the same information that you see in the image should help.

Plug-in generating a 500 response: Solution

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
```

A web or application server 500 level error can be solved by inspecting:

- The network for any performance or DNS issues
- The operating system TCP/IP layer
- To ensure that the server is not hung or busy
- To ensure that the server was properly started and open for e-business
- The `plugin-cfg.xml` **ConnectTimeout** and **ServerIOTTimeout** settings to ensure that they are not set too low
- Ensure proper working order of any firewalls or switch devices that are between the plug-in and the server
 - Verify that they allow traffic between the back-end transport ports (for example: 9080, 9443)

© Copyright IBM Corporation 2013

Figure 14-24. Plug-in generating a 500 response: Solution

WA5913.0

Notes:

This error is typically seen when the HTTP plug-in is not able to communicate with a back-end WebSphere Application Server V7.0. A WebSphere Application Server can also return it when an error is encountered while processing a request (for example, a servlet or JSP page).

The “Internal Server Error” message can be produced in one of three ways:

- In rare cases when using CGI scripts within the web server, a 500 can be produced if it has a problem with the script.
- The plug-in is unable to make a connection to any cluster member.
- The application server encountered an error when processing the request.

In the situations described above, a packet trace is useful to determine where breakdowns occur. To get the details on how to set up a packet trace for your operating system, see the website:
<http://www.ibm.com/support/docview.wss?uid=swg21175744>

HTTP plug-in 500 level generated

- Locate GET request in `http_plugin.log`:
 - [Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL:
GET /PlantsByWebSphere HTTP/1.1
- Locate the response in `http_plug-in.log`:

```
[Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL:  
lib_htresponse: htresponseRead: Reading the response: 837c74  
[Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL:  
HTTP/1.1 500 Internal Server Error  
[Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL: Date:  
Sat, 17 Mar 2009 22:01:51 GMT  
[Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL: Server:  
WebSphere Application Server/7.0  
[Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL:  
Content-Type: text/xml; charset=UTF-8  
[Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL:  
Content-Language: en-US  
[Wed Jun 27 16:27:00 2009] 0000047c 00000874 - DETAIL:  
Transfer-Encoding: chunked  
[Wed Jun 27 16:27:00 2007] 0000047c 00000874 - DETAIL:  
Connection: Close
```

© Copyright IBM Corporation 2013

Figure 14-25. HTTP plug-in 500 level generated

WA5913.0

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 and time of the failed request. The trace entries in the image 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.

WebSphere Application Server 500 response

- Search the application server `trace.log` for 500 response codes with the same date/time stamp as in `http_plug-in.log`
- Review the entire thread ID (3605eb2e) to learn more about how the request was processed

```
[4/19/09 13:00:00:670 EST] 3605eb2e WebApp d Exception  
errorCode=500  
[4/19/09 13:00:00:670 EST] 3605eb2e WebApp d Exception  
type=java.lang.IllegalStateException  
[4/19/09 13:00:00:670 EST] 3605eb2e WebApp d Exception  
message=OutputStream already obtained  
[4/19/09 13:00:00:670 EST] 3605eb2e WebApp d Found exception  
type=java.lang.IllegalStateException with location=null
```

© Copyright IBM Corporation 2013

Figure 14-26. WebSphere Application Server 500 response

WA5913.0

Notes:

This slide shows an example of the WebSphere Application Server trace that shows the 500 error code.

The trace entry in the image 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 might 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.

Solving a 500 response

- “500 Internal Server Error” responses are more difficult to resolve because their reasons are vast and sometimes complex
- If the server returns a 500 response, it is an indication that something is wrong within the application code or configuration
- For more information about debugging “500 Internal Server Error” messages, see the WebSphere Application Server support page; search on keywords “status code 500”:
 - <http://www.ibm.com/software/webservers/appserv/was/support/>
 - Numerous technotes are available with solutions to known issues and defects that are related to the 500 response code

© Copyright IBM Corporation 2013

Figure 14-27. Solving a 500 response

WA5913.0

Notes:

14.3.Troubleshooting HTTP session issues

Troubleshooting HTTP session issues



© Copyright IBM Corporation 2013

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

8.0

Figure 14-28. Troubleshooting HTTP session issues

WA5913.0

Notes:

After completing this topic, you should be able to:

- Understand the web request flow
- Identify various ways of serving a request
- Interpret web server and plug-in log files
- Enable tracing of the web server, plug-in, and web container

Troubleshooting HTTP session issues (1 of 3)

- Problem: HTTP sessions are not created or are lost between requests
- Verify that:
 1. Enable cookies is selected in the Session Manager
 2. Cookies are enabled on the browser
 3. Cookie domain specified in the Session Manager
 4. Cookie path in the Session Manager

The screenshot shows the 'Session management' configuration page. The 'Session tracking mechanism' section contains four checkboxes: 'Enable SSL ID tracking' (unchecked), 'Enable cookies' (checked), 'Enable URL rewriting' (unchecked), and 'Enable protocol switch rewriting' (unchecked). The 'Cookie domain' field is empty. The 'Cookie maximum age' section has two radio buttons: 'Current browser session' (selected) and 'Set maximum age' (unchecked). The 'Cookie path' section has two radio buttons: 'Use the context root' (unchecked) and 'Set cookie path' (checked), with a value of '/' entered.

Figure 14-29. Troubleshooting HTTP session issues (1 of 3)

WA5913.0

Notes:

A session is a series of requests to a servlet, originating from the same user at the same browser. Sessions allow applications that run in a web container to track individual users. By default, the session manager uses cookies to store the session ID on the client between requests. Unless you intend to avoid cookie-based session tracking, ensure that cookies are flowing between the application server and the browser.

Further explanation.

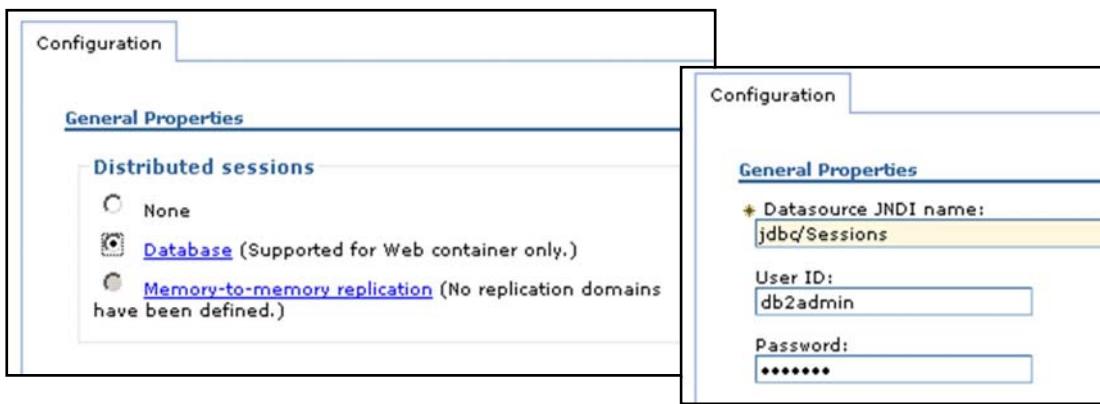
1. Under the **Session tracking Mechanism**, verify that the **Enable cookies** check box is selected.
2. Make sure that cookies are enabled on the browser you are testing from or from which your users are accessing the application.
3. Check the **Cookie domain** that is specified on the SessionManager in the **Session tracking mechanism > enable cookies** property. Click **Modify**. As an example, if the cookie domain is set to ".myCompany.com", resources should be accessed by using that domain name; for example: `http://www.myCompany.com/myapp/servlet/sessionServlet`

If the domain property is set, verify that it begins with a dot (.). Some versions of Netscape do not accept cookies if domain name does not start with a dot. Internet Explorer accepts the domain with or without a dot.

4. Check the **Cookie path** that is specified on the SessionManager. Determine whether the problem URL is hierarchically below the Cookie path specified. If not, then correct the Cookie path.

Troubleshooting HTTP session issues (2 of 3)

- Problem: HTTP sessions are not persistent
- Verify the data source
- For database-based persistence, verify these session manager persistence settings:
 - JNDI name of the data source
 - Correct userid and password to access database
 - Data source should be non-JTA



© Copyright IBM Corporation 2013

Figure 14-30. Troubleshooting HTTP session issues (2 of 3)

WA5913.0

Notes:

If your HTTP sessions are not persistent, that session data is lost when the application server restarts or is not shared across the cluster. You should consider checking the following session manager persistence settings properties:

If using session persistence, verify that Persistence is set to **Database**.

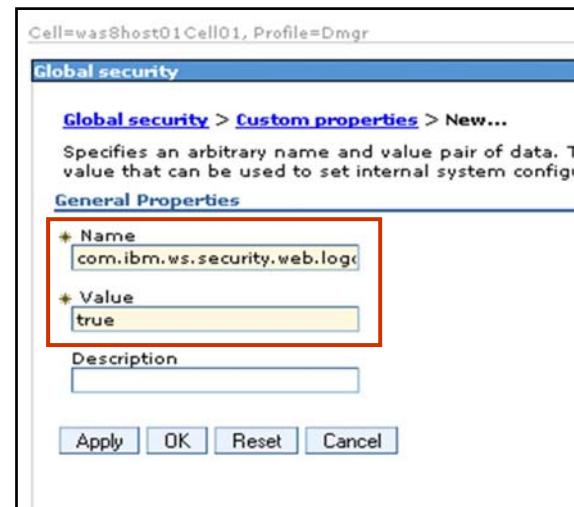
If you are using **Database-based persistence**, then follow these guidelines:

- Check the JNDI name of the data source is specified correctly on SessionManager.
- Specify the correct user ID and password for accessing the database. These settings must be checked against the properties of an existing data source in the administrative console. The session manager does not automatically create a session database for you.
- The data source should be non-JTA; for example, it should not be XA-enabled.
- Check the logs for appropriate database error messages.
- For DB2 with row sizes other than 4k, verify that the specified row size matches the DB2 page size. Verify that the table space name is specified correctly.

If you are using **memory-based persistence**, review the **Internal Replication Domains properties** of your session manager.

Troubleshooting HTTP session issues (3 of 3)

- Problem: Users are not logged out after the HTTP session timer expires
- Resolution: Complete the following steps to log out users from the application after the HTTP session expires
 - Navigate to **Security > Global security > Custom properties**
 - Click **New**
 - Enter: `com.ibm.ws.security.web.logoutOnHTTPSessionExpire`
 - Enter **true**
 - Save, resynchronize, and restart the server



© Copyright IBM Corporation 2013

Figure 14-31. Troubleshooting HTTP session issues (3 of 3)

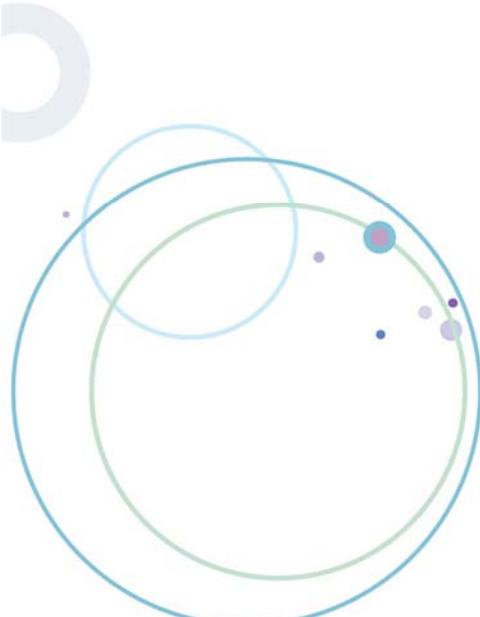
WA5913.0

Notes:

If a user logs on to an application that is served from WebSphere Application Server and sits idle longer than the specified HTTP session timeout value, the user's information is not invalidated. The user credentials stay active until LTPA token timeout occurs.

14.4.Troubleshooting dynamic cache issues

Troubleshooting dynamic cache issues



© Copyright IBM Corporation 2013

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

8.0

Figure 14-32. Troubleshooting dynamic cache issues

WA5913.0

Notes:

After completing this topic, you should be able to:

- Understand the web request flow
- Identify various ways of serving a request
- Interpret web server and plug-in log files
- Enable tracing of the web server, plug-in, and web container

Purpose of the dynamic cache

- Improve application performance by caching the output of servlets, commands, and JSP files
- Servlet caching is configured on the web container
- After invocation, the servlet caches its output
- Cache entry might contain:
 - Servlet output
 - Information about calls to other servlets
 - Information about calls to JSP files
 - Servlet metadata (timeout or priority information)

© Copyright IBM Corporation 2013

Figure 14-33. Purpose of the dynamic cache

WA5913.0

Notes:

The dynamic cache service works within an application server Java virtual machine (JVM), intercepting calls to cacheable objects. For example, it intercepts calls through a servlet service method or a command execute method. Then, it either stores the output of the object to the cache or serves the content of the object from the dynamic cache.

Troubleshooting the dynamic cache service: Servlets

Dynamic cache service troubleshooting		
Issue	Explanation	Response
Servlets not cached		<ul style="list-style-type: none"> Enable servlet caching On the web container settings page, select Enable servlet caching
Cache entries are not written to disk	When cache is full, cache entries that are written to disk and new entries added to memory cache	Verify that Disk offload is enabled on the dynamic cache service page
Servlets are not replicated or written to disk		Verify that both attributes and response are serializable
Cache is often evicted	Cache entries are evicted when the cache is full and new entries added to the cache	<ul style="list-style-type: none"> Increase cache size Enable Disk offload so entries are written to disk

© Copyright IBM Corporation 2013

Figure 14-34. Troubleshooting the dynamic cache service: Servlets

WA5913.0

Notes:

These issues are just a few of the ones that you might encounter regarding the dynamic cache service when configuring it for servlet caching. You can also configure caching for:

- Portlet fragments
- Edge Side Include
- Commands
- Web services
- JAX-RPC web services client



Cache monitor: Overview

- Web application that provides real-time view of current state of the dynamic cache
- Only way to manipulate cache data is through the cache monitor

Statistic	Value
Cache Size	2000
Used Entries	0
Cache Hits	0
Cache Misses	0
LRU Evictions	0
Explicit Removals	0
Default Priority	1
Servlet Caching Enabled	Yes
Disk Offload Enabled	No

© Copyright IBM Corporation 2013

Figure 14-35. Cache Monitor: Overview

WA5913.0

Notes:

The Cache Monitor is a web application that provides a real-time view of the current state of the dynamic cache. It is used to verify that the dynamic cache is working as expected. The Cache Monitor provides the only means to manipulate cache data.

1. Monitor cache statistics such as number of cache hits, cache misses, and number of entries in each cache instance. With this data, you can tune the cache configuration to improve the dynamic cache performance.
2. Manipulate and monitor the data that flows through the cache. Entries are distinguished because each has a unique ID string.
3. View the data offloaded to the disk. You can view the content that is copied to disk that corresponds to the view of the contents that are cached in memory for each cache instance.
4. View the cache policies, which provide rules for each cacheable object.

The Cache Monitor application is in the `<app_server_root>/installableApps` folder. The name of the application is `CacheMonitor.ear`. Install it through the administrative console or from the command line. It should be installed onto the application server that you want to monitor.

Cache monitor: Managing cache data

- Using these operations, you can manually change the state of the cache without restarting the server:
 - Remove an entry from a cache instance
 - Remove all entries for a specific dependency ID
 - Move an entry from disk to memory within a cache instance
 - Clear entire contents of the cache instance
 - Clear the contents of the disk for the cache instance

© Copyright IBM Corporation 2013

Figure 14-36. Cache Monitor: managing cache data

WA5913.0

Notes:

You can perform the following basic operations on the data in the cache without restarting the server:

- Remove an entry from a cache instance
- Remove all entries for a certain dependency ID
- Remove all entries for a certain name (template)
- Move an entry to the front of the least recently used queue to avoid removal of the cache entry
- Move an entry from the disk to the memory within a cache instance
- Clear the entire contents of the cache instance
- Clear the contents of the disk for the cache instance

14.5.Using diagnostic providers to troubleshoot web container issues

Using diagnostic providers to troubleshoot web container issues



© Copyright IBM Corporation 2013

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

8.0

Figure 14-37. Using diagnostic providers to troubleshoot web container issues

WA5913.0

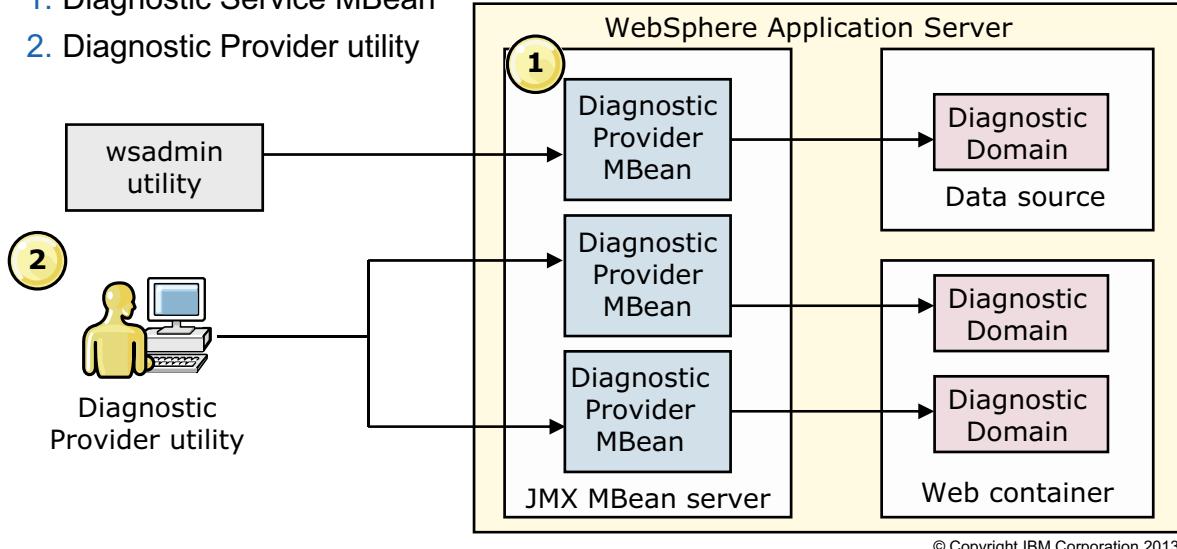
Notes:

After completing this topic, you should be able to:

- Understand the web request flow
- Identify various ways of serving a request
- Interpret web server and plug-in log files
- Enable tracing of the web server, plug-in, and web container

What is a diagnostic provider?

- Allows viewing of configuration and current state of components in an application server environment
- Gives you information to discover and diagnose system problems
- Two diagnostic provider facilities:
 - Diagnostic Service MBean
 - Diagnostic Provider utility



© Copyright IBM Corporation 2013

Figure 14-38. What is a diagnostic provider?

WA5913.0

Notes:

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 the example that is shown, the data source has one diagnostic domain and the web container has two diagnostic domains. A diagnostic domain refers to a set of classes within the component (for example, data source and web container) that share a set of diagnostic messages. Some larger components might have multiple diagnostic domains. In addition, Diagnostic Provider MBeans can also provide access to any self-diagnostic tests that are available from the diagnostic domain.

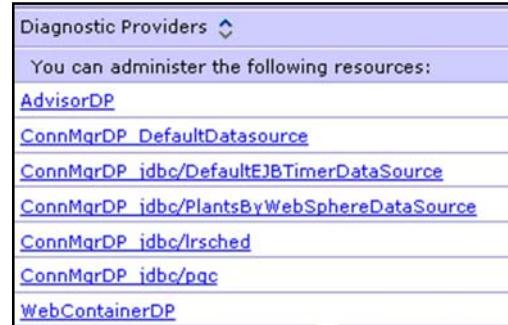


Diagnostic Provider utility

- Run and view:
 - Diagnostic self tests
 - State dumps
 - Configuration dumps



- Diagnostic providers vary depending upon the component



© Copyright IBM Corporation 2013

Figure 14-39. Diagnostic Provider utility

WA5913.0

Notes:

The Diagnostic Provider utility is included in the WebSphere Application Server administration console. 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. It provides a means to execute and view the results of configuration dumps, state dumps, and diagnostic self tests.

A **configuration dump** is an operation that you can perform on a Diagnostic Provider to list the startup or current values of the configuration attributes for the DP. The collection of attributes that are returned from this operation can be thought of as the **payload** of the configuration dump.

A **state dump** is similar to a configuration dump, but it differs in two key areas. First, a state dump displays current information about the operation of a component. An example is a connection pool. A configuration dump can show *DataSource name*, the *minConnections* (configured or current), the *maxConnections*, the *DataBase name*, and more. A state dump is likely to show the current connections in use, the high concurrent use count, the number of times the pool was expanded, and the average time between requesting a connection and returning it.

Diagnostic self tests are non-invasive operations that a Diagnostic Provider exposes. Non-invasive means that if they modify anything for the test, the conclusion of the test reverses the

modification. These tests give an administrator the option to test simple functions of a component to see whether it is able to perform them.

Step 1: Server selection topology

Step 2: Diagnostic Providers

Step 3: State specification settings

Partial results from state dump

Node	Server	Name	Value	Description
was8host01Node01	server1	ard-executionTimeoutEncoding	Null	executionTimeoutEncoding.description
was8host01Node01	server1	ard-executionTimeoutMessage	CWRDI0014E: Timed out waiting for asynchronous servlet include job to finish.	executionTimeoutMessage.description
was8host01Node01	server1	ard-executionTimeoutOverride	60000	executionTimeoutOverride.description

© Copyright IBM Corporation 2013

Figure 14-40. State dump from WebContainerDP

WA5913.0

Notes:

This example illustrates gathering a runtime state dump from the web container diagnostic provider (WebContainerDP). In step 1, you select to view state data for server1. Step 2 provides a list of all diagnostic providers available for the component, in this case, server1. After clicking WebContainerDP, the state dump results are shown in step 3. There are several options available after the state dump is produced. You can save the results to a log file for easier viewing. You can further refine the information that is returned by clicking the state specification settings button.

Configuration dump from WebContainerDP (1 of 2)

- Some configuration problems can be diagnosed from a configuration dump

Problem: Static content and HTML files do not display properly

Resolution: Verify that `fileServingEnabled` is set to true for the application in question

```
startup-vhosts-admin_host-webapps-filetransfer#filetransfer.war-
fileServingEnabled = true
```

Problem: Some types of resources display correctly, but you cannot display a servlet by its class name

Resolution: Verify that `servletServingByClassnameEnabled` is set to true

```
startup-vhosts-default_host-webapps-
DefaultApplication#DefaultWebApplication.war-
servletServingByClassnameEnabled = true
```

© Copyright IBM Corporation 2013

Figure 14-41. Configuration dump from WebContainerDP (1 of 2)

WA5913.0

Notes:

Analyzing a configuration dump can help you solve some common configuration problems. These two examples highlight issues with resources not displaying properly.

Configuration dump from WebContainerDP (2 of 2)

Problem: An installed application cannot be found via the correct context root

Resolution: Check for overlapping virtual host aliases

```
startup-vhosts-admin_host-aliases = :9062;:9045;
```

Problem: The same output is displayed when a servlet or JSP is repeatedly refreshed

Resolution: The page is most likely cached; verify that `cachingEnabled` is set to false

```
vhosts-default_host-webapps-
DefaultApplication#DefaultWebApplication.war-servlets-Snoop
Servlet-cachingEnabled = true
```

© Copyright IBM Corporation 2013

Figure 14-42. Configuration dump from WebContainerDP (2 of 2)

WA5913.0

Notes:

The first example illustrates a problem when an application cannot be found with the correct context root. The second example illustrates a page caching problem. These examples are just a few that show how you can use a configuration dump to solve common web container-related problems.

Unit summary

Having completed this unit, you should be able to:

- Enable tracing as appropriate for various components in the end-to-end flow
- Describe pinging techniques for components in the request flow
- Enable HTTP plug-in tracing
- Describe how to bypass external HTTP servers
- Describe specific web container issues
- Use the WebContainerDP diagnostic provider

© Copyright IBM Corporation 2013

Figure 14-43. Unit summary

WA5913.0

Notes:



Checkpoint questions

1. True or false: Two HTTP return codes that signal web request problems are 404 and 500.
2. True or false: Tracing of the HTTP plug-in is enabled by default.
3. True or false: If a web server's plug-in configuration file is not up to date, browser requests typically receive an HTTP Not Found error code.

© Copyright IBM Corporation 2013

Figure 14-44. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. True
2. False: An administrator must enable tracing of the HTTP plug-in.
3. True

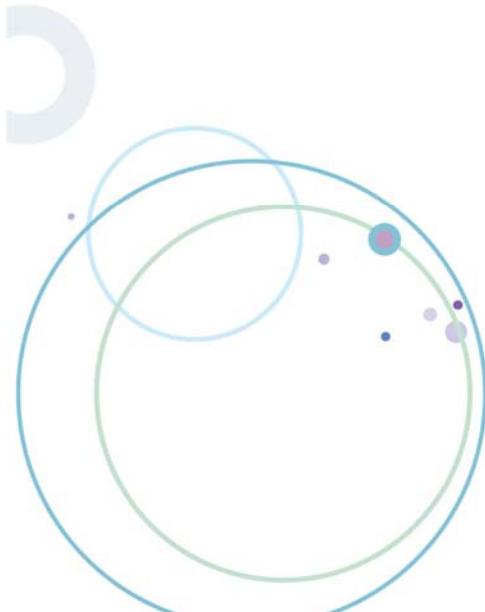
© Copyright IBM Corporation 2013

Figure 14-45. Checkpoint answers

WA5913.0

Notes:

Exercise 11



Troubleshooting request flow and web container problems

© Copyright IBM Corporation 2013

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

8.0

Figure 14-46. Exercise 11

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Configure web server logs for tracing
- Configure HTTP plug-in tracing
- Examine trace logs manually
- Use the WebContainerDP diagnostic provider
- Examine plug-in trace logs by using the Web Server Plug-in Analyzer tool

© Copyright IBM Corporation 2013

Figure 14-47. Exercise objectives

WA5913.0

Notes:

Unit 15. Default messaging provider problem determination

What this unit is about

This unit describes how to troubleshoot default messaging in WebSphere Application Server V8.5.5.

What you should be able to do

After completing this unit, you should be able to:

- Describe the components that are involved in default messaging
- Identify symptoms of default-messaging-related problems
- Collect diagnostic data
- Analyze diagnostic data and determine root causes
- Validate service integration bus (SIBus) and messaging engine configurations
- Use Java Message Service (JMS) clients to validate messaging functions

How you will check your progress

- Checkpoint
- Lab exercises

References

WebSphere Application Server V6 Problem Determination for Distributed Platforms Redbooks publication
<http://www.redbooks.ibm.com/redpieces/abstracts/SG246798.html?Open>

WebSphere Application Server V6.1 Problem Determination IBM Redpaper Collection, SG24-7461-00, Part 6 JMS problem determination

Unit objectives

After completing this unit, you should be able to:

- Describe the components that are involved in default messaging
- Identify symptoms of default-messaging-related problems
- Collect diagnostic data
- Analyze diagnostic data and determine root causes
- Validate service integration bus (SIBus) and messaging engine configurations
- Use Java Message Service (JMS) clients to validate messaging functions

© Copyright IBM Corporation 2013

Figure 15-1. Unit objectives

WA5913.0

Notes:

Topics

- WebSphere default messaging concepts
- Default messaging problem determination
- Common messaging problems
- Messaging components and trace groups

© Copyright IBM Corporation 2013

Figure 15-2. Topics

WA5913.0

Notes:

15.1. WebSphere default messaging concepts

WebSphere default messaging concepts



© Copyright IBM Corporation 2013

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

4.0 8.0

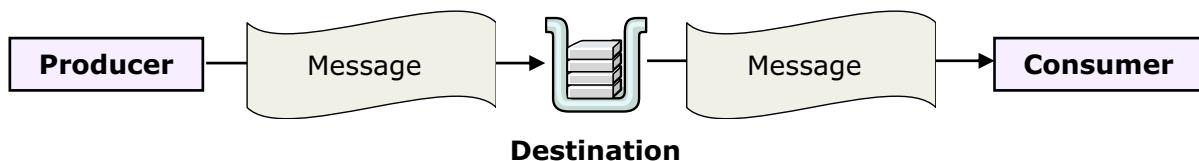
Figure 15-3. WebSphere default messaging concepts

WA5913.0

Notes:

What is JMS?

- The Java Message Service (JMS) API is used for accessing enterprise messaging systems
 - Applications asynchronously send and receive business data and events
- JMS conceptual elements include:
 - Producers – Send (put) messages to destinations
 - Consumers – Receive (get) messages from destinations
 - Destinations – Managed points for messages: A queue or a topic



© Copyright IBM Corporation 2013

Figure 15-4. What is JMS?

WA5913.0

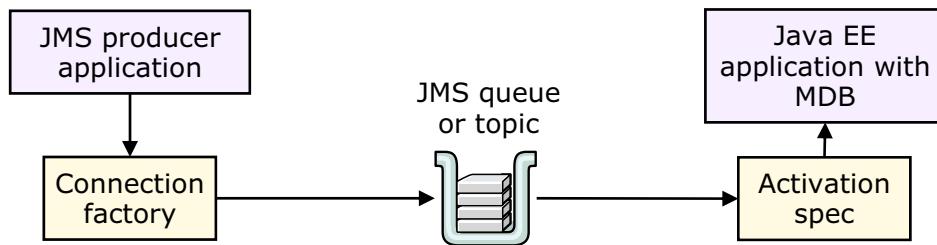
Notes:

JMS defines another protocol that WebSphere Application Server supports. Rather than a request response protocol like HTTP or RMI, JMS is an asynchronous protocol. Messages are delivered to destinations by a producer application. The producer does not wait for a response to the message. At some point in the future, a consumer application gets the message and acts on it.

JMS defines an API; it is not an implementation. JMS just defines some interfaces for applications to use. It is up to someone else to provide the implementation under those interfaces. The implementation is delivered as a **JMS Provider**.

Java EE administrative components that are used by JMS

- Destinations
 - Queues
 - Topics
- Connection factory
 - Typically used to produce messages
- Activation specification (spec)
 - Used by message-driven beans (MDBs) to consume messages



© Copyright IBM Corporation 2013

Figure 15-5. Java EE administrative components used by JMS

WA5913.0

Notes:

Producers send messages to destinations. Consumers get messages from destinations.

Destinations are managed points of communication rendezvous. Destinations can be JMS queues, JMS topics (publish/subscribe), or web service endpoints.

Another part of the JMS API is the connection factory. Typically, producer applications use a connection factory to put messages to a destination.

Message-driven beans use an Activation Spec to consume messages.

WebSphere default messaging

- Default messaging is the JMS provider that is delivered with WebSphere Application Server
- Messaging capabilities are fully integrated into the WebSphere Application Server platform
- Based on service integration bus (SIBus) technology
- Complements and extends WebSphere MQ and WebSphere Application Server
- Other WebSphere products use default messaging

© Copyright IBM Corporation 2013

Figure 15-6. WebSphere default messaging

WA5913.0

Notes:

Messaging capabilities are fully integrated in the WebSphere Application Server platform. Messaging is integrated with application server management, including security and high availability. Default messaging is fully compliant with JMS V1.1.

Because JMS is just an API definition, it is implemented in different ways by different vendors. The SIBus is the implementation that WebSphere Application Server uses to support the JMS API.

Default messaging complements and extends WebSphere MQ and WebSphere Application Server. You can share and extend messaging family capabilities and interoperate with WebSphere MQ.

Other WebSphere products use default messaging such as WebSphere Enterprise Service Bus and WebSphere Process Server. Default messaging is supported in tools by WebSphere Integration Developer.

Administrative components of default messaging

- Bus
 - Administrative concept
 - Framework for SIBus runtime components
 - Cell-scoped
- Bus member
 - Application server or cluster
 - For clusters, the entire cluster is considered a single bus member
- Destination: separate from a JMS destination, include these types:
 - Queue (associated with one bus member)
 - Topic space (associated with all bus members)
 - Others

© Copyright IBM Corporation 2013

Figure 15-7. Administrative components of default messaging

WA5913.0

Notes:

Since JMS is implemented by using the SIBus, some SIBus specific administrative components must be created to enable JMS messaging.

First, an SIBus must be created. The bus itself has no runtime behavior; it is an administrative concept that is used as a framework to manage the SIBus runtime components.

Two administrative components must be added to an SIBus. Bus members host the actual runtime services that producers and consumers connect to. Destinations are logical named places in the SIBus. SIBus destinations are not the same as JMS destinations.

The SIBus provides the following capabilities:

- Any application can exchange messages with any other application by using a *destination* to which one application sends, and from which the other application receives.
- A message-producing application, that is, a *producer*, can produce messages for a destination regardless of which messaging engine the producer uses to connect to the SIBus.
- A message-consuming application, that is, a *consumer*, can consume messages from a destination (whenever that destination is available) regardless of which messaging engine the consumer uses to connect to the SIBus.

The SIBus supports the following types of messaging:

- Sending messages synchronously (requires the consuming application to be running and reachable).
- Sending messages asynchronously (possible whether the consuming application is running or not and whether the destination is reachable). Both point-to-point and publish/subscribe messaging are supported.
- Publishing events or other notifications. The SIBus itself can also generate notification messages.

Runtime components of default messaging

- Messaging engine (ME)
 - Created when bus members are added to a bus
 - Network endpoint that producers and consumers connect to
 - Accepts JMS messages from connected producer applications
 - Delivers JMS messages to connected consumer applications
 - All MEs in a bus can communicate with each other
 - A cluster bus member can have multiple MEs
- Message store
 - Every ME requires a message store
 - Only one ME can use each message store
 - Can be file based or a database table
- SIBus service
 - Producers and consumers initially connect to the SIBus service
 - Gives producer or consumer the connection information to a specific ME
- Workload management framework (WLM)
 - Provides information that is used to route messages

© Copyright IBM Corporation 2013

Figure 15-8. Runtime components of default messaging

WA5913.0

Notes:

A messaging engine (ME) is the service that runs in a bus member that producers and consumers connect to, just as a web container accepts HTTP connections, and an EJB container accepts RMI connections. The ME accepts JMS connections.

One ME is automatically created for the application server or the cluster when defining a new SIBus member (application server or cluster). Multiple MEs can be running within the same cluster.

Within an SIBus, each ME has a unique identity, which is made up of the SIBus name and the bus member's name.

The message store is a subcomponent of the messaging engine. It is used to buffer in-flight messages and hold a number of other pieces of information (for example, records of message delivery when delivering multiple copies of a single message).

A message store can be either **persistent** or **nonpersistent**. A persistent message store can hold both persistent objects and volatile objects. Persistent objects are in a state that survives after an engine stops for any reason. Volatile objects are in a state that does not survive an ME failure, and might or might not survive an orderly shutdown of the messaging engine. In contrast, a nonpersistent message store can hold only volatile objects; it cannot hold persistent objects.

ME requires a persistent back-end data store, even for nonpersistent messages (for example, spilling).

Included is support for persistence by using Derby. Support is also included for DB2, Oracle, and Sybase to enable customers to use an RDBMS of choice. These databases should also offer higher performance than Derby. Starting with version 6.1, a file system data store is also available.

The data store preserves messages and subscriptions so that they survive if the server or messaging engine is stopped and restarted. It is also used for the overflow of the nonpersistent messages in some quality of service (QoS) options.

Every ME must have a unique message store.

The SIBus service is a separate service from an ME. An SIBus service is what producers and consumers initially connect to (called bootstrapping) so that they can get the address of a specific ME to use for sending or receiving messages.

The final runtime component that the SIBus uses is the WLM service that WebSphere Application Server provides. WLM is used to decide which ME should be used when there is more than one possible ME that can handle a connection or message.

Runtime flow for JMS messaging (1 of 4)

1 Setting up the connection factory

- a. Producer connection factory contacts SIBus service
- b. If there is no local ME, the SIBus service contacts WLM to get appropriate ME
- c. SIBus service delivers ME connection information

2 Producing

- a. Producer connection factory connects to ME
- b. Producer produces message for JMS destination
- c. If destination is not local to the producer
 1. Producer's ME contacts WLM to get the ME for the destination
 2. *Remote put* is sent to destination ME using internal queue points
- d. Destination ME associates message with destination

© Copyright IBM Corporation 2013

Figure 15-9. Runtime flow for JMS messaging (1 of 4)

WA5913.0

Notes:

Producing a message to the SIBus is a two-step process.

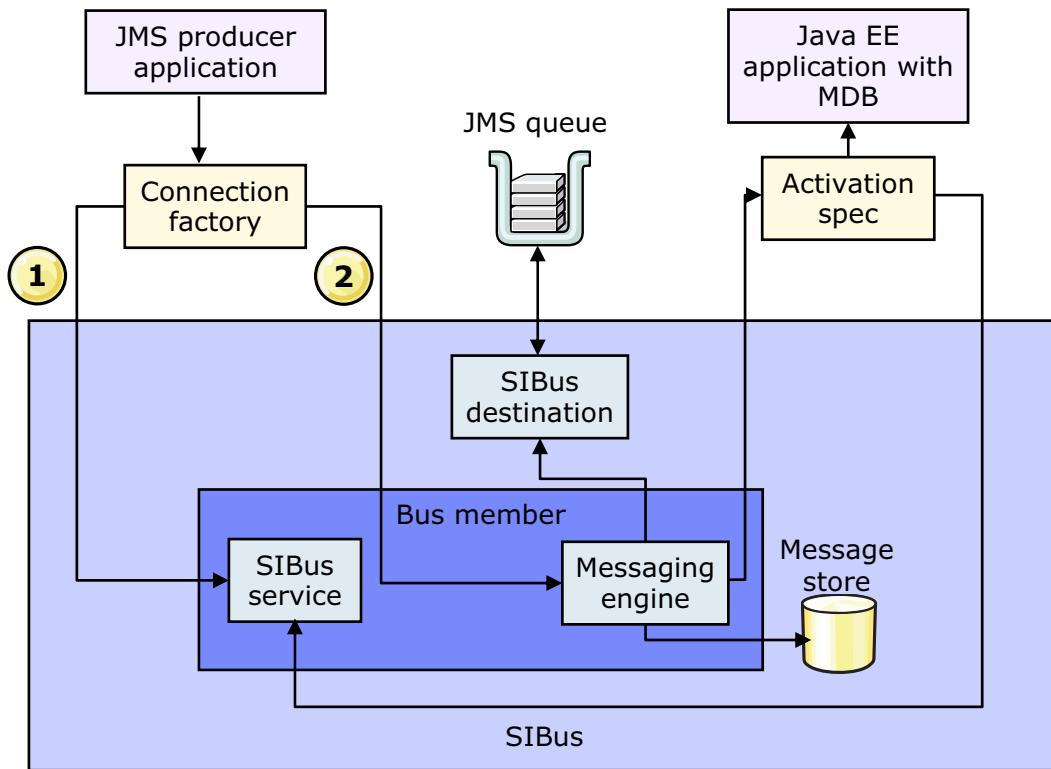
In step 1, a connection is made to an endpoint that hosts the SIBus service. Where to connect to is defined in the provider endpoints of the connection factory. If the server connected to is not running an ME, then the WLM infrastructure is used to provide the address of the ME to connect to. If the server connected to is running an ME, then the address of the local ME is provided. Step 1 must be done only once, not for every message.

In step 2, the connection factory makes a network connection to the provided ME endpoint. The connection factory produces messages to the connected ME. If the ME connected to is not the ME hosting the intended destination, internal queue points are used to route the message to the ME that is hosting the destination.

When a producing application is remote from its destination, remote queue points are used. They manage the flow of messages between the messaging engine where the destination is located, and the messaging engine to which the application is attached.

A *remote put* means that an application first sends the message to a remote queue point on the ME to which it is attached; then the message is sent to the destination.

Runtime flow for JMS messaging (2 of 4)



© Copyright IBM Corporation 2013

Figure 15-10. Runtime flow for JMS messaging (2 of 4)

WA5913.0

Notes:

1. Setting up the connection factory:

- The producer connection factory contacts the SIBus service.
- In this slide, you see that a local ME exists on the same server as the SIBus server, so the address of the local ME is returned to the producer.

2. Producing:

- The producer connection factory connects to the ME.
- The producer produces a message for the JMS destination.
- In this slide, you see that the local ME hosts the destination, so no WLM call is required to find the correct ME.

Runtime flow for JMS messaging (3 of 4)

1 Setting up the activation specification (spec)

- a. Consumer activation spec contacts SIBus service
- b. If there is no local ME, the SIBus service contacts WLM to get an appropriate ME
- c. SIBus service delivers ME connection information

2 Consuming

- a. Consumer activation spec connects to ME
- b. If destination is not local to the consumer (not suggested for performance)
 - 1. Consumer ME contacts WLM to get ME for destination
 - 2. A *remote get* is sent from the destination ME using internal queue points
- c. Consumer ME delivers message to the activation spec
- d. Consumer MDB consumes the message from the JMS destination

© Copyright IBM Corporation 2013

Figure 15-11. Runtime flow for JMS messaging (3 of 4)

WA5913.0

Notes:

Consuming a message from the SIBus is a two-step process.

In step 1, a connection is made to an endpoint that hosts the SIBus service. Where to connect to is defined in the provider endpoints of the ActivationSpec. If the server connected to is not running an ME, then the WLM infrastructure is used to provide the address of the ME to connect to. If the server connected to is running an ME, then the address of the local ME is provided. Step 1 must be done once, not for every message.

In step 2, the ActivationSpec makes a network connection to the provided ME endpoint. The ActivationSpec consumes messages from the connected ME. If the ME connected to is not the ME hosting the intended destination, internal queue points are used to route the message from the ME that is hosting the destination.

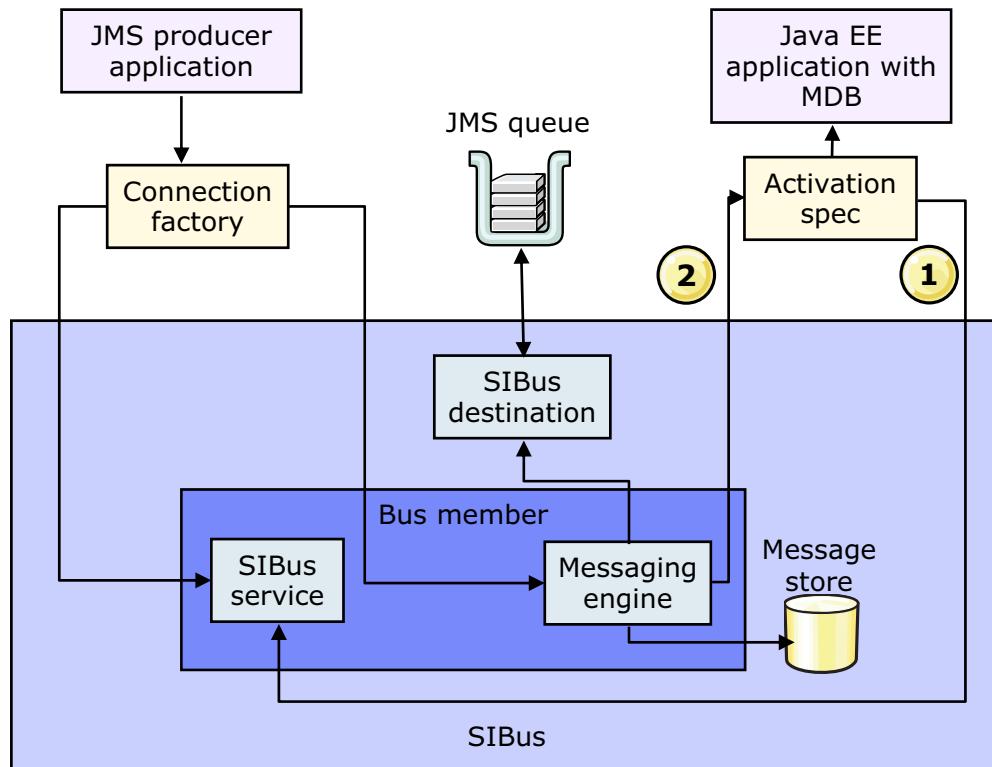
Consuming from an ME that is not hosting the destination is not a good practice. It adds unnecessary complexity, and it also results in poorer performance.

When a producing application is remote from its destination, remote queue points are used. They manage the flow of messages between the messaging engine where the destination is located and the messaging engine to which the application is attached.

When a consuming application is remote from its destination, remote queue points are also used. They manage the flow of messages between the messaging engine where the destination is located, and the messaging engine to which the application is attached.

A *remote get* means that the consumer tries to get the message from a remote queue point on the ME to which it is attached. The destination then sends the message to the remote queue so that the application can consume it.

Runtime flow for JMS messaging (4 of 4)



© Copyright IBM Corporation 2013

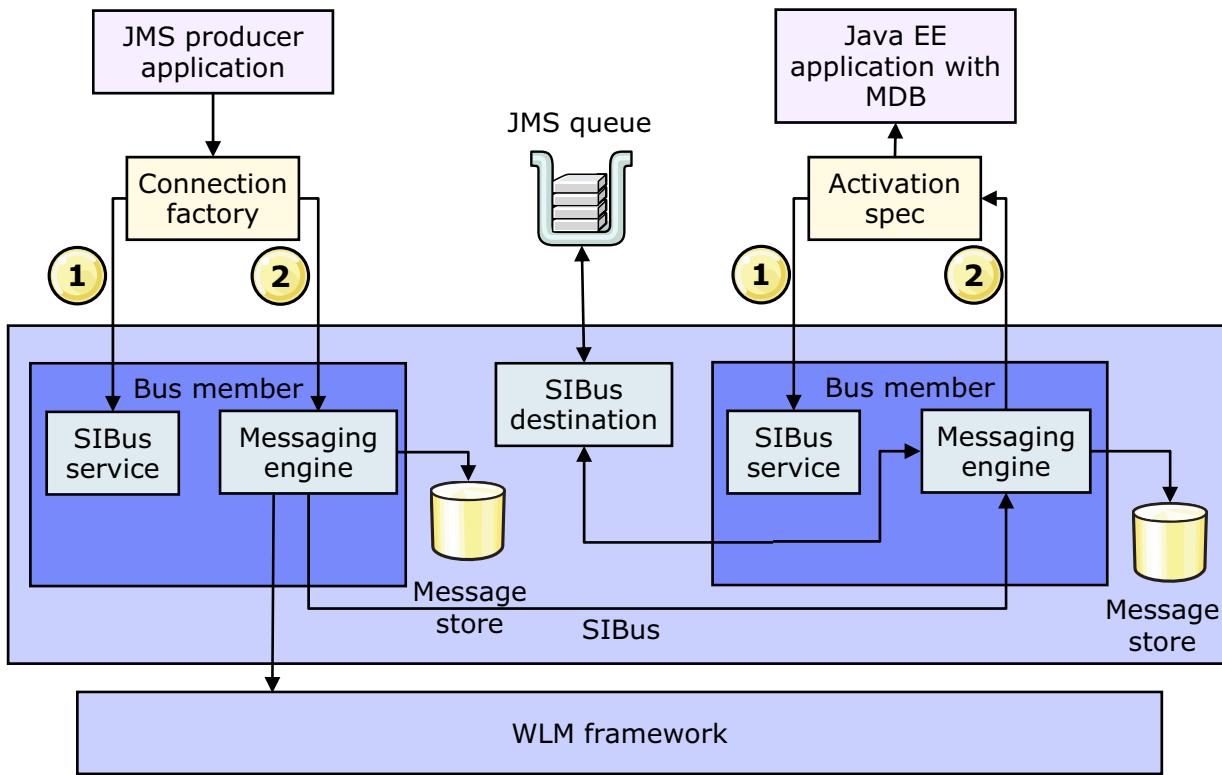
Figure 15-12. Runtime flow for JMS messaging (4 of 4)

WA5913.0

Notes:

1. Setting up the activation spec:
 - The consumer activation spec contacts the SIBus service.
 - In this slide, you see that a local ME exists on the same server as the SIBus server, so the address of the local ME is returned to the consumer.
2. Consuming:
 - The consumer activation spec connects to the ME.
 - In this slide, you see that the local ME hosts the destination, so no WLM call is required to find the correct ME.
 - The consumer ME delivers a message to the activation spec.
 - The consumer MDB consumes the message from the JMS destination.

Runtime flow for JMS messaging (more complex)



© Copyright IBM Corporation 2013

Figure 15-13. Runtime flow for JMS messaging (more complex)

WA5913.0

Notes:

In this slide, you see that the producer application bootstraps to a server that is not the bus member that hosts the intended destination. The SIBus service delivers the address of the local ME for the producer to connect to.

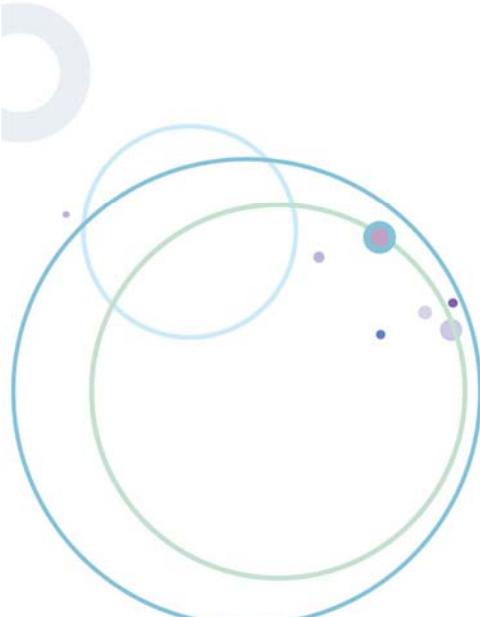
When the producer sends a message to the ME it is connected to, you find that the local ME does not host the destination. WLM must be called to determine the actual ME where the message must be sent. Internal queue points are used to send the message from the local ME to the ME that is hosting the destination.

On the consumer side, the complexity increased, not decreased, as it is suggested that consumer applications connect directly to the ME that is hosting the destination that they want to consume from.

It is possible to add considerably more complexity by connecting to SIBus services that do not have local MEs, or by connecting Activation Specs to MEs that are not hosting the intended destination.

15.2.Default messaging problem determination

Default messaging problem determination



© Copyright IBM Corporation 2013

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

4.0 8.0

Figure 15-14. Default messaging problem determination

WA5913.0

Notes:

How to approach a generic problem with WebSphere default messaging

Six steps for all messaging issues:

1. Check that the ME is in a started state



2. Check `SystemOut.log` for the exception you are trying to debug
3. Check the FFDCs for details on the specific exception
4. Search for known solutions that match your problem
5. Use MustGather document information for more detailed analysis
6. Trace and engage IBM support

© Copyright IBM Corporation 2013

Figure 15-15. How to approach a generic problem with WebSphere default messaging

WA5913.0

Notes:

The next section of this unit covers a general six-step approach for all message problems. There is an explanation in detail of each of those steps. Then, some common problems in the specific area of ME startup, message flow, and application connections are covered.

1. Check that the ME is started. If you have access to the administration panels, navigate to **Service Integration > Buses > BusName > Messaging Engine**, and then look for the green arrow that indicates 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. Search for known solutions that match your problem.
5. Use MustGather document information for more detailed analysis.
6. Trace and engage IBM support.

Check that the messaging engine is in started state

- Check the state of the ME to be sure that it started
 - Search the `SystemOut.log` file for the string “SibMessage”
- If an ME fails to start, messaging cannot function
- If the SIB service is disabled, ME cannot start
 - Enabled automatically when you add a bus member to a bus



© Copyright IBM Corporation 2013

Figure 15-16. Check that the messaging engine is in started state

WA5913.0

Notes:

Check that the ME reached a state of **started**.

Make sure that the SIB service is enabled. This service provides service integration functions.

Select the “Configuration reload enabled” option to enable the dynamic reloading of the SIB configuration files for this server.

The information that defines the configuration of service integration buses and their resources is saved in a set of configuration files. When a server starts, it uses the current information about service integration that is read from those configuration files. When a messaging engine is started, it uses the information in the server that it is running in.

If the information in the configuration files is changed while the server is running, the server must be either dynamically updated or restarted to use the updated information.

With dynamic reloading of configuration files, any updates to the configuration information are dynamically passed to the server, and therefore made available to messaging engines if they are started. You can enable dynamic reloading of configuration files for servers and for service integration buses.

If you choose not to enable dynamic reloading of configuration files, you must restart the server to pick up any changes to the configuration files.

Messaging engine startup typical messages (1 of 2)

- The ME in this case goes from Joined to Starting

```
SystemOut.log(3815): [3/15/12 11:12:46:298 EDT] 0000000f SibMessage I
[firtsbus:was8host01Node01.server1-firstbus] CWSID0016I: Messaging engine
was8host01Node01.server1-firstbus is in state Joined.
SystemOut.log(3816): [3/15/12 11:12:46:819 EDT] 00000013 SibMessage I
[firtsbus:was8host01Node01.server1-firstbus] CWSID0016I: Messaging engine
was8host01Node01.server1-firstbus is in state Starting.
```

- The ME then grabs its lock on its tables in the database

```
SystemOut.log(3905): [3/15/12 11:12:59:136 EDT] 0000001c SibMessage I
[firtsbus:was8host01Node01.server1-firstbus] CWSIS1538I: The messaging
engine, ME_UUID=E9559146F69BE94B, INC_UUID=3114b5bc38c080f6, is
attempting to obtain an exclusive lock on the data store.
SystemOut.log(3906): [3/15/12 11:12:59:176 EDT] 0000001d SibMessage I
[firtsbus:was8host01Node01.server1-firstbus] CWSIS1543I: No previous
owner was found in the messaging engines data store.
SystemOut.log(3907): [3/15/12 11:12:59:227 EDT] 0000001c SibMessage I
[firtsbus:was8host01Node01.server1-firstbus] CWSIS1537I: The messaging
engine, ME_UUID=E9559146F69BE94B, INC_UUID=3114b5bc38c080f6, has acquired
an exclusive lock on the data store.
```

© Copyright IBM Corporation 2013

Figure 15-17. Messaging engine startup typical messages (1 of 2)

WA5913.0

Notes:

A correctly running ME completes this cycle upon each startup of the application server that hosts it. Completion of this cycle is the first thing to check when debugging any WebSphere default messaging problem. Search the `SystemOut.log` for the string `SibMessage`.

Messaging engine startup typical messages (2 of 2)

- The ME then goes into a state of **Started** and is ready for action

```
SystemOut.log(3908): [3/15/12 11:13:04:704 EDT] 00000013 SibMessage      I  
[firtsbus:was8host01Node01.server1-firstbus] CWSID0016I: Messaging engine  
was8host01Node01.server1-firstbus is in state Started.
```

© Copyright IBM Corporation 2013

Figure 15-18. Messaging engine startup typical messages (2 of 2)

WA5913.0

Notes:

The ME is now in a state of started and begins doing its work. If you looked at the log for a cluster bus member that was not currently hosting an active ME, you would see only the ME in the joined state.

Check the SystemOut.log

- Search the `SystemOut.log` file for the exceptions
- Read the details of the exception stack trace
- Determine the SIB component by using information center topic:
 - Troubleshooting service integration technologies
- Use component information to help determine next steps

```
SystemErr R javax.jms.JMSException: CWSIA0241E: An exception was received
during the call to the method JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIRuntimeException: CWSIT0088E: There are
currently no messaging engines in bus msgBus running. Additional failure information:
CWSIT0103E: No messaging engine was found that matched the following parameters:
bus=msgBus, targetGroup=null, targetType=BusMember, targetSignificance=Preferred,
transportChain=InboundSecureMessaging, proximity=Bus..null
[3/28/12 0:03:19:665 EDT] 0000002f SystemErr R at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(Jm
sManagedConnectionFactoryImpl.java:195) null
[3/28/12 0:03:19:680 EDT] 0000002f SystemErr R at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(Jm
sManagedConnectionFactoryImpl.java:135) null
[3/28/12 0:03:19:680 EDT] 0000002f SystemErr R at
com.ibm.msgsender.servlet.MSGSenderServlet.sendJMSMessage(MSGSenderServlet.
java:98) null
```

© Copyright IBM Corporation 2013

Figure 15-19. Check the SystemOut.log

WA5913.0

Notes:

In the example error message on this slide, you can see that `JmsManagedConnectionFactory` threw an exception when an attempt was made to get a connection. This exception tells the tale, and it is as basic as “I just shut down the ME for this bus and it is complaining about ‘`CWSIT0088E: There are currently no messaging engines in bus msgBus running.`’”

If the exception was being reported from `sib.JMSRa`, for example, then you would know that the problem is with the JMS resource adapter. A problem with the `sibJMSRa` component would indicate a problem with a client application or MDB talking to the bus.

Check the FFDCs for details

- FFDCs report further details of an exception reported in the `SystemOut.log` file
- If installed using the defaults, on Windows you find the FFDCs in the following directory:

`<profile_root>\logs\ffdc`

```
FFDC Exception:com.ibm.websphere.sib.exception.SIRuntimeException
SourceId:com.ibm.ws.sib.api.jmsra.impl.JmsJcaManagedConnectionFactoryImpl.createManagedConnection ProbId:1
Reporter:com.ibm.ws.sib.api.jmsra.impl.JmsJcaManagedConnectionFactoryImpl@1df83d4
com.ibm.websphere.sib.exception.SIRuntimeException: CWSIT0088E: There are currently no
messaging engines in bus msgBus running. Additional failure information: CWSIT0103E: No
messaging engine was found that matched the following parameters: bus=msgBus,
targetGroup=null, targetType=BusMember, targetSignificance=Preferred,
transportChain=InboundSecureMessaging, proximity=Bus.
at
com.ibm.ws.sib.trm.client.TrmSICoreConnectionFactoryImpl.localBootstrap(TrmSICoreConnection
FactoryImpl.java:363)
at
com.ibm.ws.sib.api.jmsra.impl.JmsJcaManagedConnectionFactoryImpl.createManagedConnection(
JmsJcaManagedConnectionFactoryImpl.java:475)
```

© Copyright IBM Corporation 2013

Figure 15-20. Check the FFDCs for details

WA5913.0

Notes:

With WebSphere Application Server, there are many FFDCs. Most are not 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 time stamp with the exception report in the `SystemOut.log`. Another approach is to search the FFDC directory for the specific exception or even component name.



Perform phase 1 problem determination

- Search the IBM knowledge base for clues to a solution for the problem
 - Use IBM Support Assistant search functions
 - Check the WebSphere Application Server support site
- Specific problems:
 - ME startup problems
 - Application connection problems
 - Message flow problems

© Copyright IBM Corporation 2013

Figure 15-21. Perform phase 1 problem determination

WA5913.0

Notes:

Based on symptoms and log messages, you might begin phase 1 problem determination by searching various resources such as the WebSphere Application Server support site.

For more information, see: <http://www.ibm.com/software/webservers/appserv/was/support/>

Use MustGather document more detailed analysis

- Determine trace strings to use
 - Use SIBus MustGather documents to determine trace strings
- Reading a default messaging trace is difficult
 - SIBMessageTrace*=all Use this trace string for details of the message flow
 - Look for CWSJU messages

In trace.log of Producer ME:

```
CWSJU0002I: A producer ... sent a message with ID ... to destination
firstDestination.
CWSJU0004I: A message with ID ... is put to queue firstDestination
CWSJU0003I: A message with ID ... is committed to destination
firstDestination ...
CWSJU0021I: A message with ID ... is transmitting to messaging engine
with ID was8host01Node01.server2-firstBus ...
```

In trace.log of Consumer ME:

```
CWSJU0020I: A message with ID ... was received from the messaging
engine with ID ...
CWSJU0004I: A message with ID ... is put to queue firstDestination
CWSJU0003I: A message with ID ... is committed to destination
firstDestination ...
CWSJU0041I: A message with ID ... was delivered to a consumer ... from
destination firstDestination
```

© Copyright IBM Corporation 2013

Figure 15-22. Use MustGather document more detailed analysis

WA5913.0

Notes:

See the Service Integration Technology MustGather technote, “Collect troubleshooting data for a service integration bus (SIB) problem”:

<http://www.ibm.com/support/docview.wss?uid=swg21266769>



Engage IBM support

- To get the best support from IBM, you should prepare a clear problem statement and collect the appropriate documentation before you open a PMR
- Before you call IBM Support, prepare the following information:
 - Full problem description with details on how the problem happened, including what changed just before the problem occurred
 - At a minimum, have the `SystemOut.log` file ready for submission
 - It speeds resolution time if you have a collector tool output, and even trace, ready for submission
 - Optional: If you can re-create the problem, send in a test case that produces the problem with recreation steps

© Copyright IBM Corporation 2013

Figure 15-23. Engage IBM support

WA5913.0

Notes:

WebSphere default messaging technotes are in the same place as all other WebSphere Application Server technotes:

<http://www.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 must consult IBM Support.

Before you call IBM Support, prepare as follows:

- Make a full problem description with details on how the problem happened, including what changed just before the problem occurred.
- At least, have the `SystemOut.log` file ready for submission.
- It speeds resolution time if you have a collector tool output, and even trace, ready for submission.
- Optional: If you can re-create the problem, send in a test case that produces the problem with re-creation steps.

15.3.Common messaging problems

Common messaging problems



© Copyright IBM Corporation 2013

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

4.0 8.0

Figure 15-24. Common messaging problems

WA5913.0

Notes:

WebSphere default messaging problem solving

- Avoid problems: keep it simple
 - Use the least complex configuration that meets the required availability, scalability, and performance requirements
 - Sometimes a single clustered ME that all producers and consumers connect to is enough
- Specific problems:
 - ME startup problems
 - Application connection problems
 - Message flow problems

© Copyright IBM Corporation 2013

Figure 15-25. WebSphere default messaging problem solving

WA5913.0

Notes:

One rule of thumb to help avoid messaging problems is to avoid complexity in the SIBus configuration.

ME startup problems

- It is not always apparent at application server startup that the ME did not start
- If the messaging engine is stopped, check the log for messages
- **CWSID00016I**
 - This message tells you that the ME is stopped
 - Other messages might be present to help determine why the ME is stopped
- **CWSIS0002E**
 - This message might contain the information that tells you the cause of the problem, or you might need to scan back through the log to find more information
- **CWSID0027I**
 - This message indicates that a serious problem occurred during ME startup or restart

© Copyright IBM Corporation 2013

Figure 15-26. ME startup problems

WA5913.0

Notes:

Search the `SystemOut.log` file for `SibMessage`. This message shows the sequence of the states that the ME got into on its way to starting. As soon as it is known what state the ME got to, look at the message to see why it did not move further along.

ME startup failures: Message store problems

- Message store not available: If an ME cannot contact the data store, then the ME cannot start
 - Ensure that the database server is started correctly
- Attempt to restart the messaging engine
 - If the messaging engine does not start, restart the application server
- Test connectivity to the messaging data store database
 - This does not work if the database is secured and the authentication alias was defined at the messaging engine data store level

© Copyright IBM Corporation 2013

Figure 15-27. ME startup failures: message store problems

WA5913.0

Notes:

The message store component is responsible for writing persistent message data out to storage so that it can be recovered in the event of a failure. You might have problems when trying to interact with the data store, including SQL exceptions or difficulties with getting exclusive access to the data. This component might also generate much output if a messaging engine handles a high load of persistent messages.



ME startup failures: Message store problems

- Database message stores
 - JNDI name error when accessing the messaging data store
 - Invalid authentication alias information
 - Usually CWSIS messages
- File system message stores
 - File system not mounted
 - Wrong permissions
 - Usually CWSOM messages

Configuration

General Properties

UUID	1EA27F7891EC4E4B
* Data source JNDI name	jdbc/com.ibm.ws.sib/was8host01Node01.server3-firstBus
Schema name	IBMWSSIB
Authentication alias	(none)
<input checked="" type="checkbox"/> Create tables	
Number of tables for permanent objects	1
Number of tables for temporary objects	1

© Copyright IBM Corporation 2013

Figure 15-28. ME startup failures: message store problems

WA5913.0

Notes:

The cause of a JNDI naming problem is usually 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.

The cause of an invalid authentication alias problem is usually 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 > Bus_Name > Messaging engines > ME_Name > Data store.**

ME startup failures: Database UUID problems

- Invalid database tables
- When an ME first accesses the database, the ME registers its unique UUID in the database SIBOWNER table
 - Each ME must have its own schema
 - Point all MEs to different schemas
- If you delete and re-create an ME, it gets a new UUID, so it is not able to access the old schema
- Same problem can occur if you restore a backup of an ME that was re-created
 - Look for message CWSIS1535E
 - Solution: Drop and re-create database tables

The screenshot shows a configuration interface for a data source. The 'General Properties' section includes fields for 'UUID' (1EA27F7891EC4E4B), 'Data source JNDI name' (jdbc/com.ibm.ws.sib/was8host01Node01.server3-firstBus), 'Schema name' (IBMWSSIB), 'Authentication alias' (none), and a checked 'Create tables' checkbox. The 'Create tables' checkbox is highlighted with a red box. Below it are fields for 'Number of tables for permanent objects' (1) and 'Number of tables for temporary objects' (1). The entire configuration window has a blue border.

© Copyright IBM Corporation 2013

Figure 15-29. ME startup failures: Database UUID problems

WA5913.0

Notes:

The most common reasons for having an incorrect UUID in the data store tables are as follows. An administrator deletes an application server or removes it from the bus, and then tries to use the same database when adding a server to the bus. Or, an administrator creates a bus, deletes it, and then creates a new bus with the same name and configuration.

ME startup failures: Database lock problems

- Database tables for ME are locked
 - When an ME is running, it keeps a transaction open to the SIBOwner table (has a lock open on the table)
 - If there is a network outage, the ME can failover, but the database might not release the lock
 - The new instance of the ME cannot get the lock
- Databases use the TCP keep-alive feature that eventually recognizes that the ME is disconnected, and the database releases the lock
 - MEs keep trying to get the lock for 15 minutes
 - TCP keep-alive default is typically longer than 15 minutes
- Solution: Reduce the TCP keep-alive setting for the database server

© Copyright IBM Corporation 2013

Figure 15-30. ME startup failures: Database lock problems

WA5913.0

Notes:

See the WebSphere Support Technical Exchange presentation, “WebSphere Application Server – Service Integration Bus Messaging Engine Data Store Connectivity Problems and Solutions”:
<http://www.ibm.com/support/docview.wss?uid=swg27020333>

TCP KeepAlive is the solution to this problem. KeepAlive is a feature of TCP that has two main benefits. It can detect that a connection to a peer is no longer valid, and it can prevent a network connection from being terminated due to inactivity.

In WebSphere Application Server recovery scenarios, KeepAlive can play an important role; in particular, its ability to detect invalid connections is useful during failover scenarios. When an invalid connection is detected, the socket for that connection is cleaned up (eliminated). This cleanup is vitally important in disconnection and failover situations where rapid recovery is important. KeepAlive checks for stale sockets at certain intervals.

KeepAlive has various settings, but for the purposes of rapid recovery when database disconnections occur, you are interested in only one parameter.

KeepAlive periodically checks for broken connections (stale or orphaned sockets). By default KeepAlive checks for stale connections if the connection is idle (no packet traffic) for no less than 2

hours. As a result, if an unexpected disconnection occurs, KeepAlive will, by default, wait for 2 hours before deciding to do anything with the stale socket. What is needed is an adjustment to this 2-hour period. You can reduce the 2-hour wait to something more reasonable, such as 3 or 5 minutes. Then, the socket is cleaned up quickly, the database releases the old lock on the SIBOWNER table, and the messaging engine is able to acquire a fresh lock and resume messaging.

Application connection issues

- A client application receives a report of CWSIT0006E
- This code indicates that the bus the application is trying to connect to, is not available
 - Check for network problems
 - Check that bootstrap servers are available

© Copyright IBM Corporation 2013

Figure 15-31. Application connection issues

WA5913.0

Notes:

CWSIT0006E: It was not possible to contact any of the specified bootstrap servers.

- Explanation: The client cannot connect to the bus. This situation might be due to configuration problems, network problems, or it might be that none of the required bootstrap servers are currently available.
- Action: Ensure that the network is working correctly and that the required bootstrap servers are available. See the information center for details on configuring a connection to a bootstrap server.

Application connection issues: Service unavailable

- If a client application receives a `ServiceUnavailableException`, check the bootstrap address and port number (provider endpoint) for accuracy
 - If the provider endpoint is blank, it defaults to localhost
 - If the SIB service is not running on localhost, connection fails
 - Solution: Specify provider endpoint for a different host, or start SIB service
- Check to make sure the application server the application is connecting to, is started
 - Option is available to specify servers to connect to as “preferred” or “required”
 - If using “required” servers, make sure at least one of the required servers is available

© Copyright IBM Corporation 2013

Figure 15-32. Application connection issues: Service unavailable

WA5913.0

Notes:

Provider endpoints are specified when creating a connection factory or an ActivationSpec. If they are left blank, then the default is to connect to localhost. If the application server that runs the producer or consumer application is not also a bus member, then by default it is not going to run the SIBus service, so the connection fails. You can either explicitly specify a server where the SIBus service is running (typically any bus member), or explicitly start the SIBus service on the application server where the producer or consumer is running.

The provider endpoint can also specify target servers to connect to. If targets are specified as required, and all of the servers that host those targets are down, then the connection fails, even if other MEs are available on the bus.

Application connection issues: Unresponsive server

- You might see a CWSIJ0042E message in the log

A connection from host <host_name> has been terminated after it became unresponsive after a period of 7 seconds.

- SIB sends heartbeats across idle network connections from applications to MEs, and between MEs
- If a heartbeat fails (default wait is 7 seconds), the connection is terminated
- Applications must explicitly try the connection again
- Default heartbeat is too short
 - Long GC cycle
 - Busy server
 - Look for HMGR0152W: CPU Starvation detected in server logs
- Solution: Increase the SIB heartbeat timeout

© Copyright IBM Corporation 2013

Figure 15-33. Application connection issues: Unresponsive server

WA5913.0

Notes:

The solution is to increase the SIB heartbeat timeout. See the IBM technote, “Tuning the WebSphere Service Integration Bus Heartbeat Properties”:

<http://www.ibm.com/support/docview.wss?uid=swg21422114>

Application connection issues: MDB initialization

- You might see a CWSIV0775W message in the log: Could not start message endpoint
 - Logged when EJB container starts an application with an MDB
 - Timing issue
 - Application is initialized before SIBus completes initialization
- MDB initialization is attempted again
 - Look for CWSIV0777I: Endpoint eventually gets initialized
 - If no CWSIV0777I message, the application is misconfigured

© Copyright IBM Corporation 2013

Figure 15-34. Application connection issues: MDB initialization

WA5913.0

Notes:

If an MDB is initialized before the SIBus service completes initialization, you might get an error. The connection to the SIBus service is tried again, and should eventually succeed. If it does not, then there is a problem, which is connected to the SIBus service (see previous slides).

Message flow problems (1 of 4)

- Producing messages works fine, but the consumer does not get them
- It is possible that the ME where the queue point of the destination is hosted, is not running
 - The solution is to check to make sure the ME that is hosting the queue point is running
- Check the exception destination because messages that failed to be delivered, might end up there



© Copyright IBM Corporation 2013

Figure 15-35. Message flow problems (1 of 4)

WA5913.0

Notes:

Messages can get stuck on internal queue points if the ME hosting the destination is not running. If consumer applications are not receiving messages, check the exception destination to make sure that the messages are being sent correctly.

Message flow problems (2 of 4)

- There is one exception destination (queue) for each messaging engine
- The messages can be viewed the same way that you would view messages on any other destination

Buses > firstBus > Destinations

A bus destination is defined on a service integration bus, and is hosted by one or more locations within the bus. Applications can attach to the destination as producers, consumers, or both to exchange messages.

Preferences

New... Delete Mediate Unmediate

Select Identifier Bus Type Description Mediation

You can administer the following resources:

		firstBus	Topic space		
<input type="checkbox"/>	Default.Topic.Space	firstBus	Topic space		
<input type="checkbox"/>	SYSTEM.Exception.Destination.was8host01Node01.server1-firstBus	firstBus	Queue		
<input type="checkbox"/>	SYSTEM.Exception.Destination.was8host01Node01.server2-firstBus	firstBus	Queue		
<input type="checkbox"/>	firstDestination	firstBus	Queue		

Total 4

© Copyright IBM Corporation 2013

Figure 15-36. Message flow problems (2 of 4)

WA5913.0

Notes:

Exception destinations are always queues.

Any messages sent to the exception destinations can be viewed in the administrative console.



Message flow problems (3 of 4)

- If destination ME is started, check the message queues
 - WebSphere default messaging uses internal queues to move messages between applications and MEs and between MEs
 - From the administrative console, select **Buses > Bus_name > Messaging engines > ME_name > Runtime tab > Remote queue points > Destination_name > Outbound messages**
 - Check for communication errors between the ME with messages on its remote queue point, and the destination ME

© Copyright IBM Corporation 2013

Figure 15-37. Message flow problems (3 of 4)

WA5913.0

Notes:

If consumer applications are not receiving messages, it is possible to check the internal queue points to see where messages are being held.

Message flow problems (4 of 4)

- Viewing the internal queue points in the administrative console

Buses > firstBus > Messaging engines > was8host01Node01.server1-firstBus > Remote queue points > firstDestination > Outbound messages > Messages

A snapshot of the current outbound messages for the message point.

Preferences

Select	Sequence ID	API message ID	Time produced	State
<input type="checkbox"/>	2500004	5000009	3/15/12 9:06:19 AM	Pending Send
<input type="checkbox"/>	2500005	5000010	3/15/12 9:06:19 AM	Pending Send
<input type="checkbox"/>	2500006	5000011	3/15/12 9:06:19 AM	Pending Send
<input type="checkbox"/>	2500007	5000012	3/15/12 9:06:19 AM	Pending Send
<input type="checkbox"/>	2500008	5000013	3/15/12 9:06:19 AM	Pending Send
Total 5				

© Copyright IBM Corporation 2013

Figure 15-38. Message flow problems (4 of 4)

WA5913.0

Notes:

Here is a view of an internal queue point with messages that are waiting to be sent to the final destination.



Message flow problems: Messages not consumed

- MDBs are not processing messages
- Check the message states and transaction IDs for the destination queue point

Buses

Buses > firstBus > Destinations > firstDestination > Queue points > firstDestination@was8host01Node01.server2-firstBus > Messages

The messages on the message point.

+ Preferences

Delete	Delete all	Refresh	Jump to page...	
Select	Position	System message ID	State	Transaction ID
You can administer the following resources:				
<input type="checkbox"/>	1	745150264D1979E5_2000017	Removing	0000004e0000004e
<input type="checkbox"/>	2	745150264D1979E5_2000018	Available	
<input type="checkbox"/>	3	745150264D1979E5_2000019	Available	
<input type="checkbox"/>	4	745150264D1979E5_2000020	Available	
<input type="checkbox"/>	5	745150264D1979E5_2000021	Available	
<input type="checkbox"/>	6	745150264D1979E5_2000022	Available	
Total 6				

© Copyright IBM Corporation 2013

Figure 15-39. Message flow problems: Messages not consumed

WA5913.0

Notes:

By looking at the queue point for the actual destination, you can see the state of the messages: Available or Removing.

Message flow problems: States (1 of 2)

- In locked state with a transaction ID
 - Application is taking a long time to complete (possibly deadlocked)
 - A transaction became in-doubt, ME waiting for directions from the transaction manager
- In locked state without a transaction ID
 - Message was passed to a consumer, consumer did not yet complete
 - Message was passed to a consumer, consumer completes, SIB did not try to remove the message yet
 - The message was passed over the network to a client application preemptively (up to one per client with read-ahead disabled, and many more if read-ahead is enabled) – the client did not yet consume it

© Copyright IBM Corporation 2013

Figure 15-40. Message flow problems: States (1 of 2)

WA5913.0

Notes:

In many cases, messages are being delivered correctly, but the consuming application is either slow, deadlocked, or misconfigured.

Message flow problems: States (2 of 2)

- Not in locked state
 - Nobody is listening for messages on this destination
 - The match criteria of all the consumers for this destination do not match the message
 - All consumers for this destination are pointing at a different partition of the destination in a WLM environment
 - All consumers are maxed out processing messages and cannot except new ones (normally happens when there is a mix of locked and unlocked messages)
- If message IDs are changing, the application is slow
- If none of the above situations apply
 - Turn on tracing for ME of destination
 - Get Java cores for servers that are running the consumer applications
 - Contact IBM support

© Copyright IBM Corporation 2013

Figure 15-41. Message flow problems: States (2 of 2)

WA5913.0

Notes:

Check the message IDs, lock state, and transaction IDs of messages to help determine why messages are not being consumed. If none of the situations that are described in these two slides apply, then you turn on tracing and do further investigation.

Message flow problems: Lost messages

- Messages seem to be getting lost
 - WebSphere default messaging lose messages by design, depending on quality of service
 - In general, the higher the quality of service, the lower the performance

© Copyright IBM Corporation 2013

Figure 15-42. Message flow problems: Lost messages

WA5913.0

Notes:

If the MDB fails to handle the message that is delivered to it, the ME attempts to redeliver the message. The number of times the ME attempts to redeliver the message to the MDB is determined from the maximum failed deliveries parameter of the destination. As soon as the maximum number of retries is exhausted, the message is delivered to the defined exception destination. If no exception destination is defined, then the maximum failed deliveries parameter is ignored, resulting in an infinite loop when the message is not deliverable to the MDB.

Message flow problems: Lost messages

- 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 might be working as designed

Quality of service	ME failure	ME shutdown	Comms failure	High load
Best effort nonpersistent	Discarded	Discarded	May be discarded	May be discarded
Express nonpersistent	Discarded	Discarded	May be discarded	Preserved
Reliable nonpersistent	Discarded	Discarded	Preserved	Preserved
Reliable persistent	May be discarded	Preserved	Preserved	Preserved
Assured persistent	Preserved	Preserved	Preserved	Preserved

© Copyright IBM Corporation 2013

Figure 15-43. Message flow problems: Lost messages

WA5913.0

Notes:

On the chart, **Discarded** means that the message is going to be discarded, **May be discarded** means that the message might be discarded, and **Preserved** means that the message is not going to be discarded.

15.4. Messaging components and trace groups

Messaging components and trace groups



© Copyright IBM Corporation 2013

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

4.0 8.0

Figure 15-44. Messaging components and trace groups

WA5913.0

Notes:



Components and their trace groups (1 of 3)

- **Sib.jms**
 - WebSphere default messaging JMS provider code
 - The trace group is **SIBJms*=all**
- **Sib.comms**
 - 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 that fit into the WebSphere Application Server administrative framework
 - Trace group is **SIBAdmin*=all**

© Copyright IBM Corporation 2013

Figure 15-45. Components and their trace groups (1 of 3)

WA5913.0

Notes:

In the next several slides, the default messaging trace components are identified.

Sib.jms is the WebSphere default messaging JMS provider code.

Sib.comms is the communications code in WebSphere default messaging. It handles both client-to-ME, and ME-to-ME communication.

Sib.admin is the component that provides the WebSphere default messaging administrative objects, for example buses or destinations, which fit into the WebSphere Application Server administrative framework.

Components and their trace groups (2 of 3)

- **Sib.mfp**

- Handles message formatting and parsing
- It changes the format of a message that is 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 is **SIBMfp*=all**

- **Sib.messagestore**

- Manages 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 is **SIBMessageStore=all**

- **Sib.processor**

- Defines the ME; it is the “heart and soul” of WebSphere default messaging
- Trace is **SIBProcessor=all**

- **Sib.security**

- Security piece of WebSphere default messaging
- Trace is **SIBSecurity=all**

© Copyright IBM Corporation 2013

Figure 15-46. Components and their trace groups (2 of 3)

WA5913.0

Notes:

- Sib.mfp = “message formatting and parsing”
- Sib.mfp is the part of default messaging that does message formatting and parsing. It changes the format of a message that is 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.
- Sib.messagestore is the part of default messaging that does the SQL calls to the database. It internally does all of the XA work but does not use XA when communicating with the database.
- Sib.processor is the code that defines the ME. It is the “heart and soul” of default messaging.
- Sib.security is the security piece of default messaging. The WebSphere Application Server Object Authority Manager (OAM) does all of the work in the area of security, so this component interfaces only with the WebSphere Application Server OAM.

Components and their trace groups (3 of 3)

- **Sib.mediations**
 - Does the mediation handler work
 - The trace is **SIBMediations=all**
- **Sib.psb**
 - Publish/subscribe bridge
 - It handles publish/subscribe messaging over MQLink
 - The trace group is **SIBPsb=all**
- **Sib.trm**
 - The topology routing manager; it relies heavily on the workload manager (WLM)
 - The WLM is responsible for deciding what MEs run where
 - The WLM then tells the SIBTrm what to do
 - **WLM*=all** and **SIBTrm=all**
- **Sib.webservices**
 - Does all the web service tasks
 - There is no trace group for this component
 - You need to specify a trace string
 - At the very least, you need to trace **com.ibm.ws.sib.webservices.***

© Copyright IBM Corporation 2013

Figure 15-47. Components and their trace groups (3 of 3)

WA5913.0

Notes:

- Sib.psb = “publish/subscribe bridge.”
- Sib.trm = “topology routing manager.”
- Sib.mediations is the code that does the mediation handler work.
- Sib.psb is the publish/subscribe bridge. It handles publish/subscribe messaging over the MQLink.
- Sib.trm is the topology routing manager. It relies heavily on the WebSphere application component that is called the workload manager. The workload manager is responsible for deciding what messaging engines run where. The workload manager then tells the sib.trm what to do. The workload manager also decides what shared queue gets a message in a workload balancing cluster environment. It is common in Sib.trm problems to need the WebSphere Application Server workload manager trace as well.
- Sib.webservices is the component that does all the web service tasks.



Tracing for WebSphere default messaging (1 of 3)

- If you trace `SIB*=all`, you trace every component, every class, and every method



© Copyright IBM Corporation 2013

Figure 15-48. Tracing for WebSphere default messaging (1 of 3)

WA5913.0

Notes:

If you turn on `SIB*=all` tracing, you must increase the number of historic files as this trace is verbose. You should expect a decrease in performance.

Tracing for WebSphere default messaging (2 of 3)

- Shows details of the message flow
 - SIBMessageTrace*=all
- The following trace groups are the most verbose; combining several of these trace groups can impact performance:
 - SIBMessageStore=all – does the interaction with the database; it does the actual SQL calls to the DB
 - SIBProcessor=all – ME doing messaging
 - SIBCommunications=all – handles ME-to-ME and application-to-ME communication
 - SibMessage*=all – follows a message through the bus from producer to consumer
 - SIBMfp=all – does the message formatting and parsing

© Copyright IBM Corporation 2013

Figure 15-49. Tracing for WebSphere default messaging (2 of 3)

WA5913.0

Notes:

Using two or more of the listed trace groups might affect performance in a negative way, sometimes to the point of not allowing the problem that you are trying to trace to happen.



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.log` to decide what packages to trace
 - Consult the MustGather pages at the following link:
<http://www.ibm.com/support/docview.wss?rs=171&uid=swg21266769>
 - Consult IBM Support

© Copyright IBM Corporation 2013

Figure 15-50. Tracing for WebSphere default messaging (3 of 3)

WA5913.0

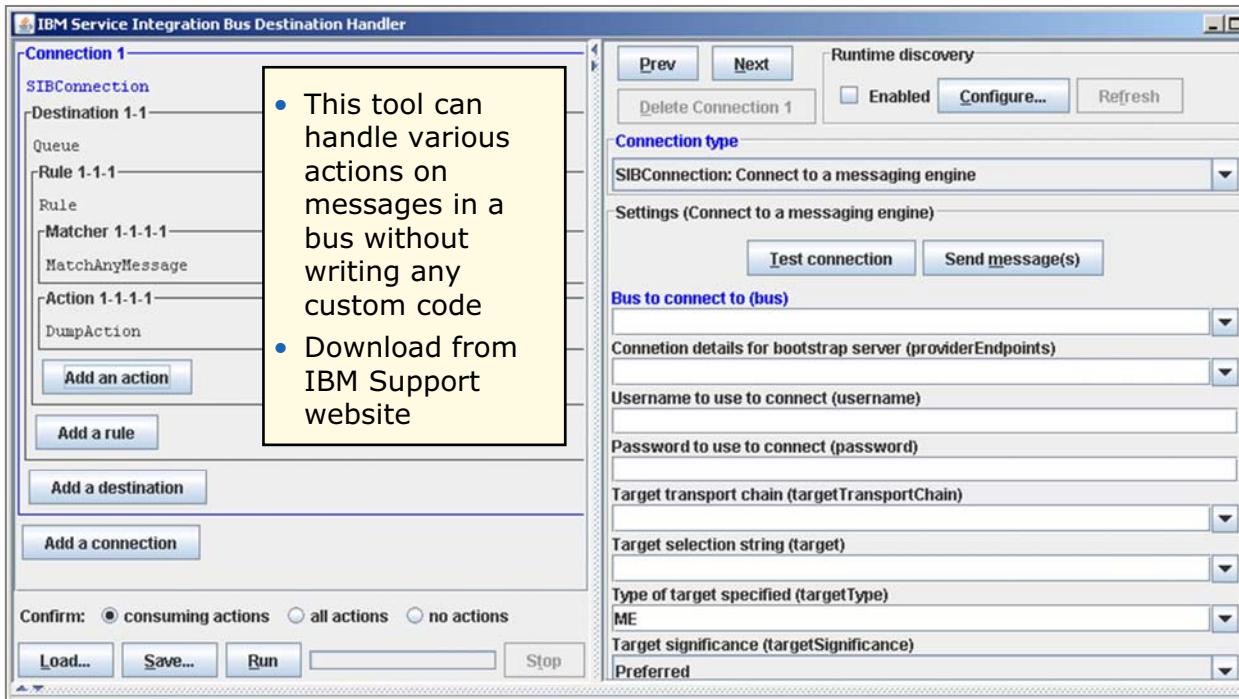
Notes:

Looking at the stack trace of an exception tells 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 good chance of your capturing the cause of the problem in trace.

Review the Redbooks publication, *WebSphere Application Server V6.1 Problem Determination IBM Redpaper Collection*, SG24-7461-00, Part 6: “JMS problem determination.”



Service Integration Bus Destination Handler (SIBDH)



© Copyright IBM Corporation 2013

Figure 15-51. Service Integration Bus Destination Handler (SIBDH)

WA5913.0

Notes:

This tool is the default messaging provider for WebSphere Application Server V6.0, V6.1, V7.0, V8.0, V8.5, and V8.5.5.

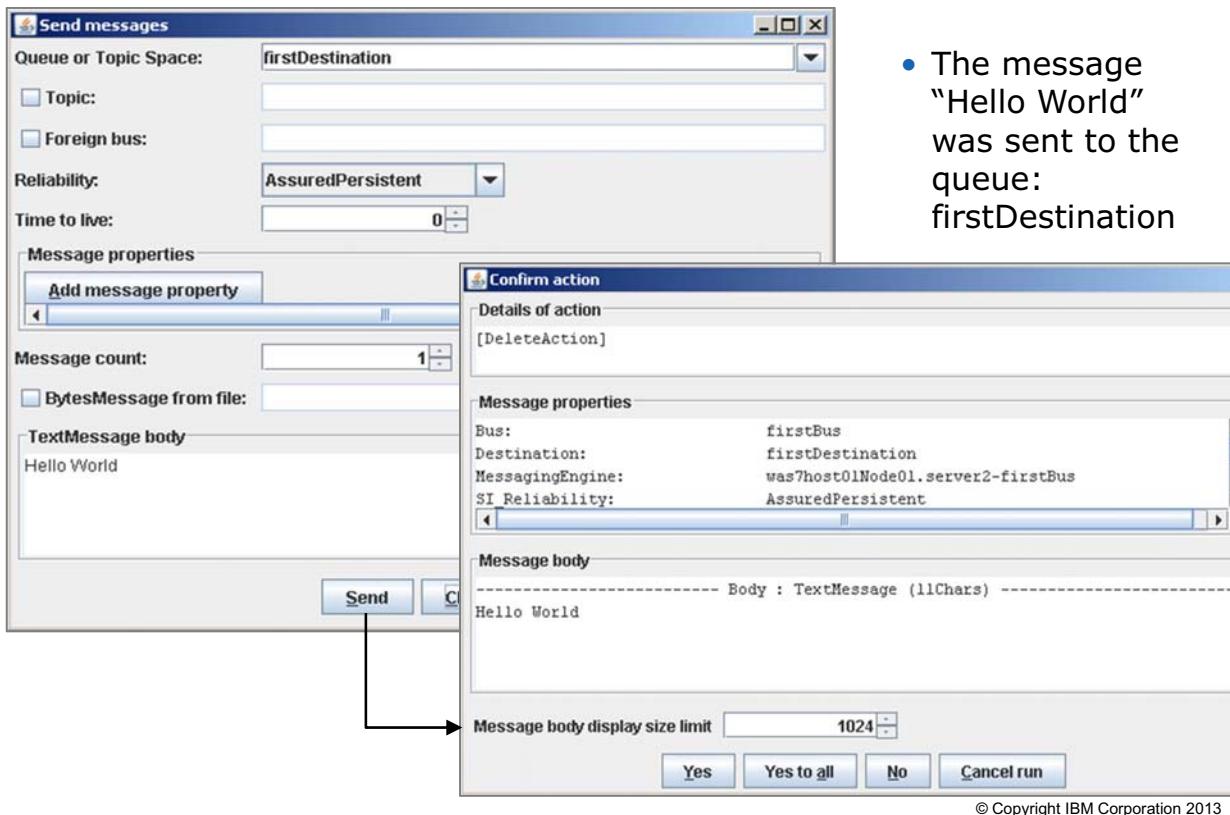
With this tool, you can perform various actions on messages in a service integration bus without writing any custom code.

These actions can be run one time for an individual task. The tool can also be deployed as a WebSphere Application Server scheduler task to regularly check the contents of an exception destination and provide appropriate handling of messages that are based on the set of rules configured.

This tool can be downloaded at:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg24021439>

Using SIBDH to send and delete a message



© Copyright IBM Corporation 2013

Figure 15-52. Using SIBDH to send and delete a message

WA5913.0

Notes:

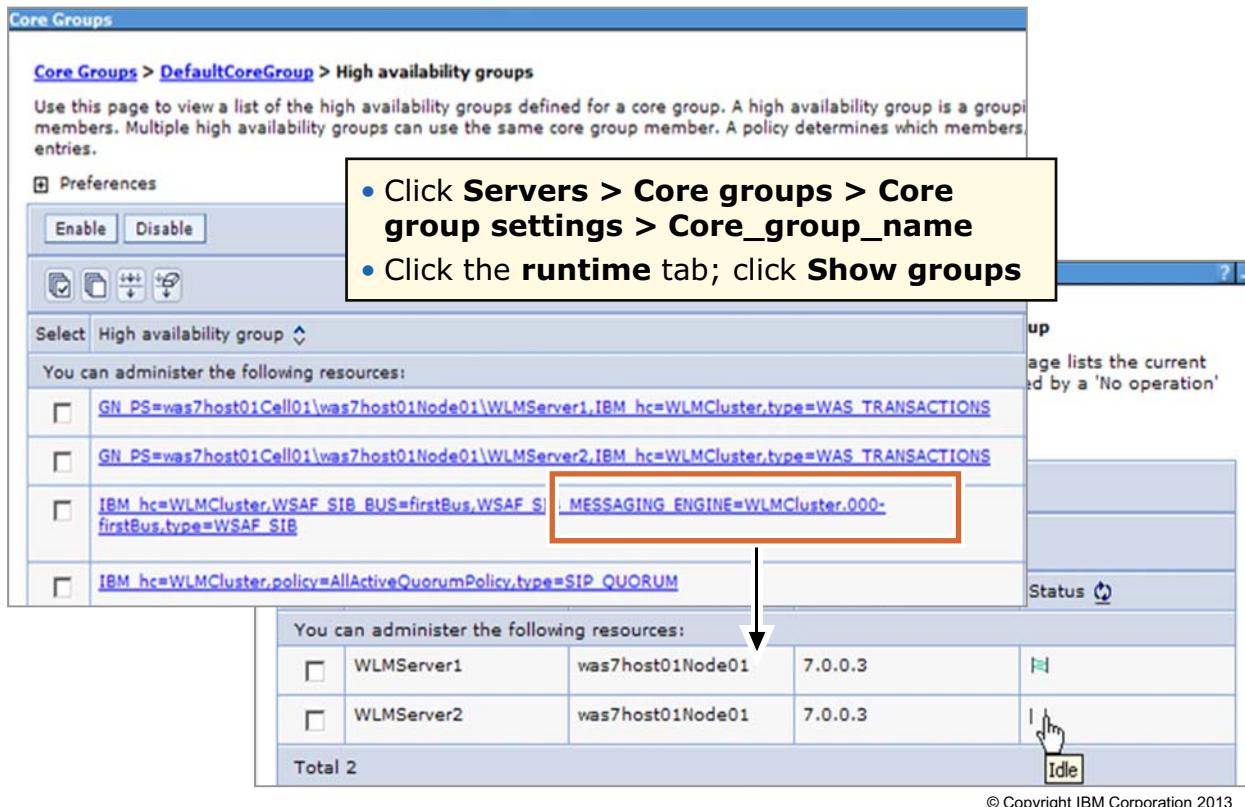
Here is a list of actions that the tool can perform, with example scenarios for their use:

- Printing a textual representation of the properties and body of messages
- Searching the contents of a queue for an individual message or group of messages
- Examining the contents of a message that an application cannot process
- Gathering a snapshot of the contents of a queue to send to an application developer for investigation
- Moving messages to a different queue point
- Handling a case where a message is sent to the wrong queue point
- Moving messages from a queue point where no application is available to consume them
- Moving messages to a different bus
- Moving messages that can never be processed from an exception destination to a permanent failure destination
- Copying messages to a different destination

- Copying messages from a production bus to a test bus for investigation
- Resubmitting messages from an exception destination back to the exception problem destination
- Resubmitting messages after restarting a database or other system that is required to process them
- Dumping a copy of the body and properties of messages to files
- Examining the body of a message with a tool that requires the binary contents
- Creating a copy of a set of messages, which an application sent, to use multiple times in testing
- Restoring messages from a file to a destination
- Loading a set of messages onto a destination as the input for a test
- Sending an email to an administrator that contains properties from messages
- Configuring an exception destination handler to inform an administrator when certain conditions occur

 WebSphere Education 

Viewing ME status from the administrative console



Core Groups

Core Groups > DefaultCoreGroup > High availability groups

Use this page to view a list of the high availability groups defined for a core group. A high availability group is a group of members. Multiple high availability groups can use the same core group member. A policy determines which members are in each group.

Preferences

Enable Disable

Select High availability group

You can administer the following resources:

<input type="checkbox"/> GN_PS=was7host01Cell01\was7host01Node01\WLMServer1,IBM_hc=WLMCluster,type=WAS_TRANSACTIONS	
<input type="checkbox"/> GN_PS=was7host01Cell01\was7host01Node01\WLMServer2,IBM_hc=WLMCluster,type=WAS_TRANSACTIONS	
<input type="checkbox"/> IBM_hc=WLMCluster,WSAF_SIB_BUS=firstBus,WSAF_SIB	MESSAGING ENGINE=WLMCluster.000-firstBus,type=WSAF_SIB
<input type="checkbox"/> IBM_hc=WLMCluster,policy=AllActiveQuorumPolicy,type=SIP_QUORUM	

Status

Idle

© Copyright IBM Corporation 2013

Figure 15-53. Viewing ME status from the administrative console

WA5913.0

Notes:

A core group is a grouping of WebSphere Application Server cell processes. A core group can contain stand-alone servers, cluster members, node agents, and the deployment manager. A core group must contain at least one node agent or the deployment manager. A core group must be empty before it can be deleted. Connected core groups are core groups that can communicate with each other. Access point groups must be defined to establish communication between core groups.

When a cluster of application servers is a member of a bus, one or more cluster members can host a messaging engine. A high availability core group is created for each cluster bus member to ensure messaging engine high availability. From the administrative console, you can see the members of the core group and the status of their messaging engine (active or idle).

Unit summary

Having completed this unit, you should be able to:

- Describe the components that are involved in default messaging
- Identify symptoms of default-messaging-related problems
- Collect diagnostic data
- Analyze diagnostic data and determine root causes
- Validate service integration bus (SIBus) and messaging engine configurations
- Use Java Message Service (JMS) clients to validate messaging functions

© Copyright IBM Corporation 2013

Figure 15-54. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. True or false: Service integration bus members can be application servers or clusters.
2. True or false: Log messages that are related to default messaging can be easily found in the SystemOut.log file by searching for "SibMessage".
3. True or false: An administrator must always start the messaging engine (ME) after the application server starts.

© Copyright IBM Corporation 2013

Figure 15-55. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.



Checkpoint answers

1. True
2. True
3. False: The messaging engine starts during the startup of the application server.

© Copyright IBM Corporation 2013

Figure 15-56. Checkpoint answers

WA5913.0

Notes:

Exercise 12



Troubleshooting WebSphere
default messaging

© Copyright IBM Corporation 2013

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

4.0 8.0

Figure 15-57. Exercise 12

WA5913.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Identify default messaging symptoms
- Examine log files for messaging-related errors
- Configure the tracing of messaging components
- Analyze trace logs to determine root causes
- Use JMS clients to verify messaging functions
- Use the IBM Service Integration Bus Destination Handler

© Copyright IBM Corporation 2013

Figure 15-58. Exercise objectives

WA5913.0

Notes:

Unit 16. WebSphere installation problems when using IBM Installation Manager

What this unit is about

This unit explains problems that can occur during installation and updating when using the Installation Manager.

What you should be able to do

After completing this unit, you should be able to:

- Identify common installation problems
- Locate and examine relevant installation logs
- Recover from a failed installation
- Search for relevant version information and prerequisite levels
- Identify problems that are associated with applying maintenance updates

How you will check your progress

- Checkpoint questions

References

MustGather: Installation Manager issues for installing and updating
WebSphere Application Server V8.0 and V8.5

<http://www.ibm.com/support/docview.wss?uid=swg21497417>

Unit objectives

After completing this unit, you should be able to:

- Identify common installation problems
- Locate and examine relevant installation logs
- Recover from a failed installation
- Search for relevant version information and prerequisite levels
- Identify problems that are associated with applying maintenance updates

© Copyright IBM Corporation 2013

Figure 16-1. Unit objectives

WA5913.0

Notes:

Topics

- Introduction to IBM Installation Manager
- How to view Installation Manager logs
- Installation problem determination
- Failure of the install, update, and uninstall processes
- Profile creation problems
- Centralized Installation Manager (CIM)
- Update and maintenance

© Copyright IBM Corporation 2013

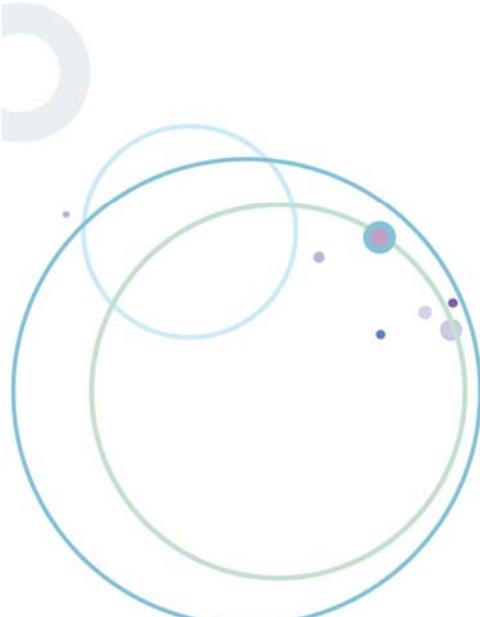
Figure 16-2. Topics

WA5913.0

Notes:

16.1. Introduction to IBM Installation Manager

Introduction to IBM Installation Manager



© Copyright IBM Corporation 2013

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

8.0

Figure 16-3. Introduction to IBM Installation Manager

WA5913.0

Notes:



Installation Manager

- IBM Installation Manager is an application that simplifies downloading, installing, updating, and uninstalling many IBM software products from a single user interface
- A single instance of IBM Installation Manager can manage the product lifecycle for any IBM Software brand
 - Provides a full graphical user interface for distributed platforms (excluding z/OS and IBM i systems)
 - Provides a utility for all platforms to run install operations silently, by using either command-line options or a response file
 - Operates on the local system where IBM products are installed, and can pull repositories from a remote system

© Copyright IBM Corporation 2013

Figure 16-4. Installation Manager

WA5913.0

Notes:

A single instance of IBM Installation Manager can manage the product lifecycle for any IBM Installation Manager based product from WebSphere, Rational, Lotus, and any other brand within IBM.

The IBM Installation Manager full product installation lifecycle management consists of product installation, adding or removing features, uninstallation, and full service management that includes installing and uninstalling interim fixes and applying and rolling back service fix packs.

IBM Installation Manager provides a full graphical user interface for distributed platforms, excluding z/OS and IBM i operation systems. It also provides a command-line utility for all platforms to perform installation operations silently.

InstallShield Multiplatform versus Installation Manager

- InstallShield Multiplatform was the former installation technology for WebSphere V7.0, V6, and V5
 - Install, uninstall and update processes required different programs
 - Each offering required a separate installation program
 - Fixpack in .pak format and installed with Update Installer (UPI)
 - Obtaining feature pack or fix pack downloads is separate process
- IBM Installation Manager for WebSphere V8
 - Single interface to install, update, and uninstall
 - Use online or local repositories (.zip format)
 - Ability to download packages that are integrated into Installation Manager
 - Single Installation Manager instance can manage multiple products
 - Replaces dedicated installers, uninstallers, and maintenance tools
 - Simplifies product maintenance

© Copyright IBM Corporation 2013

Figure 16-5. InstallShield Multiplatform versus Installation Manager

WA5913.0

Notes:

This slide explains key differences between InstallShield Multiplatform (ISMP) and Installation Manager.

IBM Installation Manager was available in version 7.0 but was used only to install feature packs.

Installation Manager: GUI mode (1 of 2)

- Started by using the IBMIM command in <IM_root>/eclipse



© Copyright IBM Corporation 2013

Figure 16-6. Installation Manager: GUI mode (1 of 2)

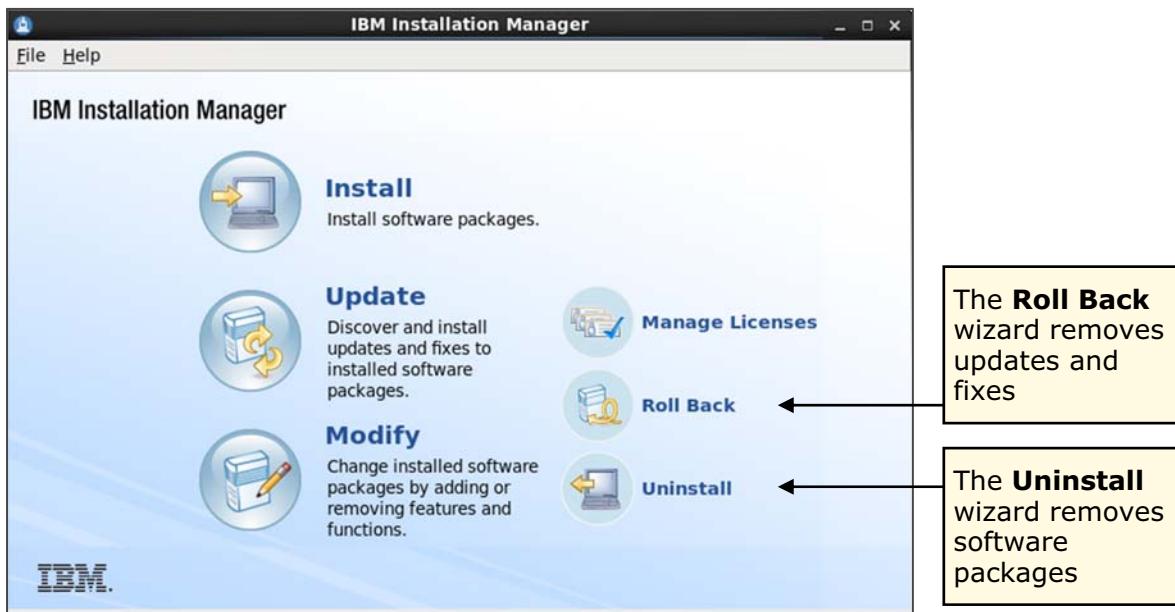
WA5913.0

Notes:

Wizards guide you through the steps that you must take to install, modify, update, roll back, or uninstall your IBM products. Use Installation Manager to install individual software packages on your local computer, or with the IBM Packaging Utility to install software for an enterprise.



Installation Manager: GUI mode (2 of 2)



© Copyright IBM Corporation 2013

Figure 16-7. Installation Manager: GUI mode (2 of 2)

WA5913.0

Notes:

Use the rollback feature to remove an update that you applied to a product package. When you roll back a package, Installation Manager uninstalls the updated resources and reinstalls the resources from the previous version. You can roll back only one version level at a time.

You can use the uninstall wizard to uninstall packages that you installed with Installation Manager. You must log in with a user account that has the same privileges as the account that was used to install the packages that you want to uninstall. You can uninstall packages from a single package group, or you can uninstall all installed packages from every package group.

Installation Manager: Silent installation

- Silent installation is done by using a response file
 - A sample response file is available from the information center
 - A response file can also be recorded
- Response file is in xml format
- Same command (such as `imcl`) is used for installing, uninstalling, updating, or rolling back
 - Need only to change options in the response file to run different operations
- A single response file can run multiple offerings install
- Sample command:

```
imcl -acceptLicense input /usr/installv8.xml -log /tmp/log.xml -sP
```

© Copyright IBM Corporation 2013

Figure 16-8. Installation Manager: Silent installation

WA5913.0

Notes:

The silent installation response file is available in the information center and also can be recorded on any machine where Installation Manager can be run in graphical mode. The file should be edited for use on machines where Installation Manager is being run in silent mode.

The command to run the response file remains the same. The same response file can be used to install multiple offerings.

Installation Manager: Command line

- The common `imcl` command options that are used are:

Command options	Function
install	To install offerings
uninstall	To uninstall offerings or interim fixes
listAvailablePackages	Find the offerings available in repositories
listAvailableFixes	List interim fixes available for installation in repository for offering
listInstalledPackages	List packages that the Installation Manager installs
Version	Version information of Installation Manager
Rollback	To roll back to previous installed fix pack
listInstallationDirectories	Directory locations of installed product

- All commands are in `<IM_root>/eclipse/tools`

© Copyright IBM Corporation 2013

Figure 16-9. Installation Manager: Command line

WA5913.0

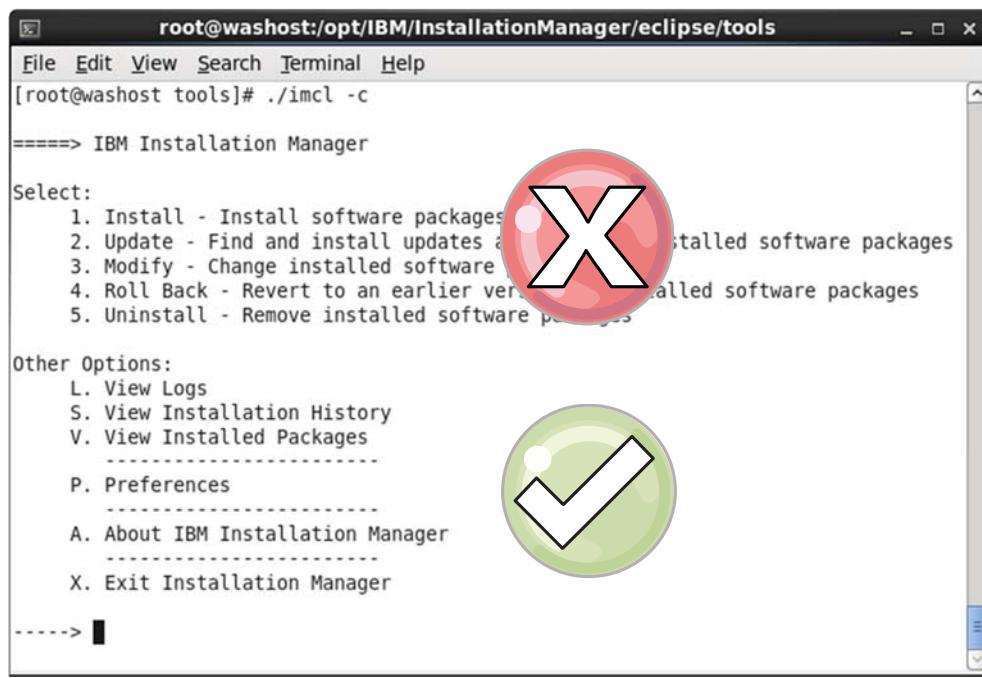
Notes:

The command-line option can also be used to install. All options are specified on the command line. This method is best suited to users who want to script the installation. Most commands use options that are listed on the slide. For more details, review the information center:

http://pic.dhe.ibm.com/infocenter/install/v1r6/index.jsp?topic=%2Fcom.ibm.cic.commandline.doc%2Ftopics%2Fr_tools_imcl.html

Installation Manager: Console mode

- WebSphere Application Server Version 8 offerings do not support use of console mode for install, update, modify, or uninstall options



© Copyright IBM Corporation 2013

Figure 16-10. Installation Manager: Console mode

WA5913.0

Notes:

The `imcl` console option is not supported for install, update, and uninstall options in WebSphere V8 offerings. But `imcl -c` can be used to view logs or installation history and change preferences on a machine where the graphical option is not available. Options beside the check mark can be used only with WebSphere.



IBM Installation Manager repositories

- An IBM Installation Manager repository contains the full content that is required to install on various operating systems
- Repositories are collections of files in a directory structure that contain product installation metadata and files
- Repository topologies can be generalized into three categories:
 - **Public repository (web repository)**: Publicly accessible using a URL, typically on ibm.com
 - **Local repository**: Used by a single system and not shared with others
 - **Enterprise repository**: Created in-house, typically exists behind the firewall and systems within the enterprise intranet access it

© Copyright IBM Corporation 2013

Figure 16-11. IBM Installation Manager repositories

WA5913.0

Notes:

The repository contains one or multiple product offerings that have both metadata and actual payload for the offering.

The offering metadata describes the name, supported platforms, optional features, and prerequisites for the product.

Normally, an IBM Installation Manager repository contains the full content that is required to install on various platforms or operating systems.

The repository can be referenced from different IBM Installation Managers on different machines.

The repository topologies can be generalized in three categories:

- A public repository, which is accessible by using a URL with an Internet connection
- An enterprise repository that typically exists behind the firewall and is accessible only to multiple machines within the enterprise intranet
- A local repository that a single user uses and does not share with others

IBM Packaging Utility can be used to copy offerings from one repository to another to host enterprise repositories.

IBM Packaging Utility can also split a huge repository into platform-specific repositories as required. Since the repository contains both binary code and text-based XML files, it must be transferred in binary mode to prevent any corruption when using file transfer tools.



IBM Packaging Utility

- IBM Packaging Utility is available to copy offerings from one repository to another and retrieve offerings from remote repositories and can:
 - Generate a new repository that contains one or more product repositories
The new repositories can be made available to your own organization by using a web server or FTP server
 - Extract offerings for a single operating system from a large repository, producing a smaller “platform-scoped” repository

© Copyright IBM Corporation 2013

Figure 16-12. IBM Packaging Utility

WA5913.0

Notes:

You can use the Packaging Utility for the following tasks:

- Generate a new repository for packages.
- Copy multiple packages to one repository.
- Copy multiple versions of a product to one repository. Users point to the same repository to update installed products.
- Save disk space when multiple packages that share components are added to one repository. Only one copy of the shared components is kept in the repository.
- Delete packages that are no longer needed.
- Create a repository to install packages over HTTP.
- Copy packages from disk installation images or IBM repositories to a repository that is on an internal server or a local computer.

Important directories: Agent data location

- `appDataLocation`, also called Installation Manager Agent Data Directory, or `IM_DATA` directory
- Installation Manager uses this location to store state and history about the installations it manages
- The agent data location for a particular operating system varies depending on who does the installation
 - Administrator or root
 - Non-administrator or non-root
 - Group

© Copyright IBM Corporation 2013

Figure 16-13. Important directories: Agent data location

WA5913.0

Notes:

Next, you look at `appDataLocation`, also known as the `IM_DATA` or `IMAgent Data` directory.

This directory is the one that Installation Manager uses to track data that is associated with installed products like state of each product, history of operations, and logs.

Changing any content in this directory is not supported.

Default agent data location

Operating system	Administrator installation	Non-administrator installation	Group installation
Windows 2008 and Windows 7	C:\ProgramData\IBM\InstallationManager	C:\Users\user\ApplicationData\Roaming\IBM\InstallationManager	Not available
Linux and UNIX	/var/ibm/InstallationManager	/user_home/var/ibm/InstallationManager	/user_home/var/ibm/Installation Manager_Group

© Copyright IBM Corporation 2013

Figure 16-14. Default agent data location

WA5913.0

Notes:

This directory is the one that Installation Manager uses to track data that is associated with installed products like state of each product, history of operations, and logs.

Changing any content in this directory is not supported. The default IM_DATA locations and details on contents are listed on the slide.

Contents of agent data

Folder or file	Information
adapters	Folder where adapters save rollback information
bundles	Contains package bundles
.settings	Preferences
histories	Previous installations and uninstalls
installRegistry.xml	Known profiles and installed packages
installRegistry	Metadata for installed offerings
installed.xml	Installation locations and installed packages
license	Empty for products that do not use Installation Manager licensing
logs	Agent log files
p2	Eclipse p2 profile metadata
pluginState	Eclipse plugin state and error log file
temp	Temporary files that are used during installation
uninstall	Executable file that is used for uninstalling packages

© Copyright IBM Corporation 2013

Figure 16-15. Contents of agent data

WA5913.0

Notes:

Changing the content, files, or directories in the agent data location directory or subdirectories is not supported. Changing the contents of the agent data location can damage the installation data. This damage can prevent you from modifying, updating, rolling back, or uninstalling installed packages and require you to uninstall packages and Installation Manager.

Important directories: Agent data location

- When Installation Manager is installed, the location of agent data cannot be changed
- If the directory is lost or corrupted, future updates to the product can be affected
 - Products run fine but cannot be updated or uninstalled
 - Recommended to back up this directory when WebSphere backup is taken
- The default location can be changed by installing the Installation Manager with the `-dL` option
- For example, a graphical install:
 - `./userinst -dL /usr/IBM/AppData` (non-root)
 - `./install -dL /opt/IBM/AppData` (root)
- The user can use different `appDataLocation` for each product by specifying the `-dataLocation` (or `-dL`) option when starting Installation Manager, and then should use the same option on subsequent invocations
 - User must maintain record of data agent that is used for each installation

© Copyright IBM Corporation 2013

Figure 16-16. Important directories: Agent data location

WA5913.0

Notes:

The data in this directory is important. If this directory becomes lost or corrupted, then it might not be possible to install fix packs or even uninstall products.

Taking backups of this directory is a good practice.

You can direct Installation Manager to use a different directory by using the `-dataLocation` or `-dL` parameter during installation of Installation Manager for all offerings that Installation Manager installs to use the specified directory.

For each offering, a different DataLocation can be used, but users must be aware of each directory as Installation Manager by default uses the data agent that is specified during installation of Installation Manager. If you choose to use a different DataLocation, then the same location must be used every time for this WebSphere Application Server installation.

Important directories: Shared directory

- The shared resources directory is used:
 - To store artifacts that more than one product use
 - As a “staging area” during installation
 - All new artifacts are first transferred to the shared resources directory
 - This process involves validating bytes by using MD5 checksum to ensure that all files are available and not corrupted
 - As a cache for rollback, without connecting to a repository
- Location is selected the first time that a package is installed and cannot be changed or moved
 - All other offerings that are installed with the same Installation Manager use the same IMShared
- Multiple products share this directory
- Choose a location with reasonable free disk space
- Contents of this directory should not be deleted to conserve file system space
 - Tips on reducing disk space are covered in later section

© Copyright IBM Corporation 2013

Figure 16-17. Important directories: Shared directory

WA5913.0

Notes:

The Shared Resources Directory acts as a cache of data that is related to repositories. It contains files that multiple packages and product installations can share. This directory's location can be selected only the first time that you install a package with the IBM Installation Manager.

After the shared data location is set, it cannot be changed by using a response file or the graphical interface.

For best results, select a file system with ample available space. This directory grows as more products and packages are installed. The contents of this directory should not be deleted to conserve file system space. Options to save on disk space (avoid saving files for rollback) is explained in later section of this presentation.

It is a good practice to take a backup of this directory when taking other operating system backups.

Contents of this directory should not be deleted to conserve file system space. Tips on reducing disk space are covered in a later section.

16.2.How to view Installation Manager logs

How to view Installation Manager logs



© Copyright IBM Corporation 2013

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

8.0

Figure 16-18. How to view Installation Manager logs

WA5913.0

Notes:

Installation Manager logs and history (1 of 2)

- Logs for all events are logged in <AGENT_DATA>/logs directory
- History of all operations is under <AGENT_DATA>/history
- Logs and history can be viewed by using graphical mode
- When running silent or command line, add -sP and -log options so any errors during install are written to the log

```
./imcl input wasresponses.xml -acceptLicense -sP -log log855.xml
 25%      50%      75%      100%
-----|-----|-----|-----|
.....
```

Installed com.ibm.websphere.ND.v85_8.5.5000.20130514_1044 to the
C:\WebSphere\AppServer8550 directory.

- For a successful installation, log855.xml contains:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
</result>
```

- Any errors or warnings are logged in the specified xml file

© Copyright IBM Corporation 2013

Figure 16-19. Installation Manager logs and history (1 of 2)

WA5913.0

Notes:

All operations in Installation Manager are logged under logs directory that is located under AGENT_DATA. There is also a history directory, which has the history of all events that Installation Manager performs.

The -log options can be used to create log during installation. The -sP option can be used to show the status of installation during silent or command-line installations as shown in the example.

Logs, history and Installed Offerings can be viewed through Installation Manager in graphical mode or locally on the machine.

```
<?xml version="1.0" encoding="UTF-8"?>
<result>
<warning uid="CRIMA1002W">
```

The following repositories are not connected:

```
-/usr/IBM-repositories/WAS85
.
</warning>
</result>
```


Installation Manager logs and history (2 of 2)

- In graphical mode, logs can be viewed by using Installation Manager

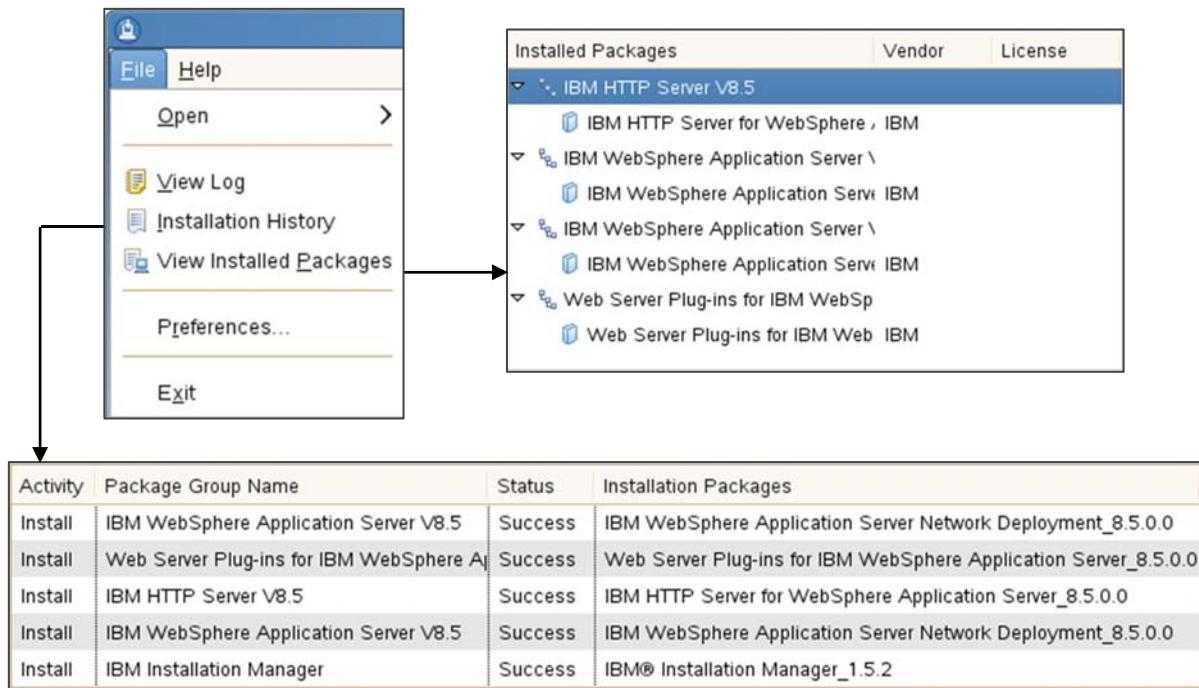


Figure 16-20. Installation Manager logs and history (2 of 2)

WA5913.0

Notes:

Using Installation Manager in graphical mode, you can view the Installed Packages, Installation History, and Logs as shown here.

Logs: Systems without graphical interface

- If the graphical option is not available on the host system, then agent data can be copied to any system for review by using local Installation Manager or web browser
 - Command can be used to create a compressed file of agent data:
`imcl exportInstallData IMDATA.zip`
 - Default locations of agent data are mentioned under Important Directories section
- In silent mode `-log <location and name of Log file>` option can be used
 - The `-log` option creates logs for the current operation
 - Any errors or warnings are written to the log specified, but the default logging still continues
- To view logs and history on a different host system, enter the command:
`IBMIM -dL <Path where zip file is extracted>`

© Copyright IBM Corporation 2013

Figure 16-21. Logs: Systems without graphical interface

WA5913.0

Notes:

If graphical mode cannot be used, then the data agent directory can be moved to a machine where you can use Installation Manager or the browser to view logs.

To view logs and data in Installation Manager, start Installation Manager with the `IBMIM -dL` option and point to the Extracted Data Agent directory.



Logs: View history by using a web browser

- From <Agent Data>/histories, open index.xml in a web browser

Install Location Name	History File
IBM Support Assistant	IBM Support Assistant/history.xml
WebSphere Customization Toolbox V8.5	WebSphere Customization Toolbox V8.5/history.xml
IBM Installation Manager	IBM Installation Manager/history.xml
Web Server Plug-ins for IBM WebSphere Application Server V8.5	Web Server Plug-ins for IBM WebSphere Application Server V8.5/history.xml
IBM HTTP Server V8.5	IBM HTTP Server V8.5/history.xml
IBM WebSphere Application Server V8.5	IBM WebSphere Application Server V8.5/history.xml

- Each history file can be opened for more details about the installation
 - Start and end times, status (success, failure), package, summary of features

© Copyright IBM Corporation 2013

Figure 16-22. Logs: View history by using a web browser

WA5913.0

Notes:

To view History of installation in the local web browser, open the index.xml file under the AGENT_DATA/histories directory in the web browser. Then, view the history of installation. Get the time stamp of activity for which you want to view logs and then view the same time in the logs directory.



Logs: View logs by using a web browser (1 of 2)

- From <Agent Data>/logs, open index.xml in a web browser

Date	Time	Entries	Errors	Warnings	Log File
2013-05-29	16:05:51	22	1	2	20130529_1605.xml
	16:06:50	22	1	2	20130529_1606.xml
	16:26:25	22	1	2	20130529_1626.xml
	20:55:25	76	0	0	20130529_2055.xml
	20:56:59	476	0	3	20130529_2056.xml
2013-05-31	16:26:22	121	0	101	20130531_1626.xml
2013-06-04	10:41:46	23	0	4	20130604_1041.xml
	10:44:27	49	0	30	20130604_1044.xml

© Copyright IBM Corporation 2013

Figure 16-23. Logs: View logs by using a web browser (1 of 2)

WA5913.0

Notes:

Open the index.xml file under AGENT_DATA/logs for review.

Pick the log that corresponds to your install attempt. You can view the time stamps from the histories directory to find the failure attempts and open logs that correspond to that time.



Logs: View logs by using a web browser (2 of 2)

- Click any log file <time_stamp>.xml to view the details

	Level	Message ID	Time	Message
1	INFO		00:00.23	No log properties file found in: /tmp/ciclogs_root
2	INFO		00:00.23	Initial log file: /tmp/ciclogs_root/20130621_1406.xml
3	INFO		00:03.63	OS Windowing System: gtk 2.18.9
4	INFO		00:03.74	No log properties file found in: /tmp/ciclogs_root
5	NOTE	CRIMC1005I	00:03.74	Log File: /var/ibm/InstallationManager/logs/20130621_1406.xml
6	INFO		00:03.77	No log properties file found in: /var/ibm/InstallationManager/logs
7	INFO		00:03.77	Starting Installation Manager.
8	INFO		00:03.77	osgi.os:linux, osgi.arch:x86_64, 64bit
9	INFO		00:03.77	os.name:Linux, osgi.nl:en_US, java.version:1.6.0

© Copyright IBM Corporation 2013

Figure 16-24. Logs: View logs by using a web browser (2 of 2)

WA5913.0

Notes:

The Install log provides a complete record of the events that take place during the installation process.



View all installed products (1 of 3)

- Open `installed.xml` under the `<Agent Data>` directory in a web browser

A screenshot of a Mozilla Firefox browser window. The title bar reads "IBM Installation Manager - Installed Offerings - Mozilla Firefox". The address bar shows "file:///var.ibm/installationManager/installed.xml". The main content area displays the following text:

**IBM Installation Manager -
Installed Offerings**

**IBM® Installation Manager Version 1.6.2
(1.6.2000.20130301_2248)**

Installation Directory:	/opt/IBM/InstallationManager/eclipse
Architecture:	64-bit

© Copyright IBM Corporation 2013

Figure 16-25. View all installed products (1 of 3)

WA5913.0

Notes:

Make sure to open files under directory agent data directories only. This image shows the different values in the `installed.xml` file.

View all installed products (2 of 3)

- IBM HTTP Server V8.5

Package Group		IBM HTTP Server V8.5
Name:		
Package Group Installation		
Directory:		/opt/IBM/HTTPServer
Package Group Eclipse IDE:		/opt/IBM/HTTPServer
Package Group Translations:		en
Package Group Architecture:		32-bit
Packages		Features
IBM HTTP Server for WebSphere Application Server Version 8.5.5.0 (8.5.5000.20130514_1044) Repository /usr/IBM-repositories_GA /WAS_V8.5.5_SUPPL		<ul style="list-style-type: none"> ◦ IBM HTTP Server 64-bit with Java, Version 6

© Copyright IBM Corporation 2013

Figure 16-26. View all installed products (2 of 3)

WA5913.0

Notes:

Information about installed products includes installation directory, packages, and features.



View all installed products (3 of 3)

- IBM WebSphere Application Server V8.5 is the profile ID
 - Must be unique per installation
- Fixpack level and any interim fix installed details
- Installed features and JDK bit level

IBM WebSphere Application Server V8.5	
Package Group Name: Package Group Installation Directory: Package Group Eclipse IDE: Package Group Translations: Package Group Architecture:	/opt/IBM/WebSphere/AppServer /opt/IBM/WebSphere/AppServer en 32-bit
Packages	Features
IBM WebSphere SDK Java Technology Edition (Optional) Version 7.0.4.1 (7.0.4001.20130510_2103) Repository <code>/usr/IBM-repositories_GA/WS_SDK_JAVA_TEV7.0</code>	
IBM WebSphere Application Server Network Deployment Version 8.5.5.0 (8.5.5000.20130514_1044) Repository <code>/usr/IBM-repositories_GA/WAS_ND_V8.5.5</code>	<ul style="list-style-type: none"> ◦ IBM 64-bit WebSphere SDK for Java ◦ EJBDeploy tool for pre-EJB 3.0 modules ◦ Embeddable EJB container ◦ Sample applications ◦ Stand-alone thin clients and resource adapters

© Copyright IBM Corporation 2013

Figure 16-27. View all installed products (3 of 3)

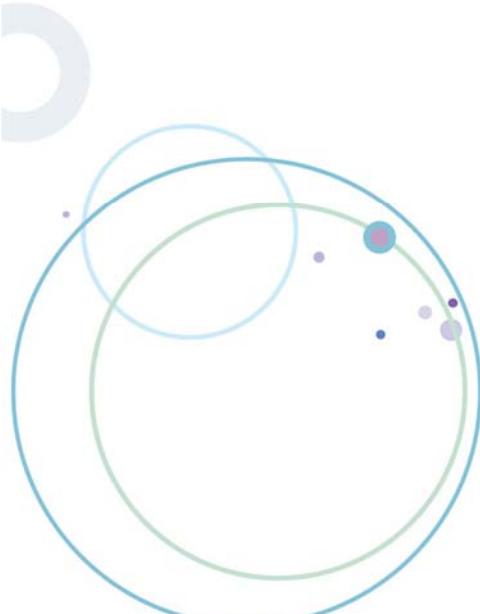
WA5913.0

Notes:

The installation log also shows fix pack levels, any installed interim fixes, and the JDK version and bit level.

16.3. Installation problem determination

Installation problem determination



© Copyright IBM Corporation 2013

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

8.0

Figure 16-28. Installation problem determination

WA5913.0

Notes:

Installation problem determination

- Evaluate the high-level symptoms to determine whether 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

Note: Make sure that you read and meet the hardware and software prerequisites

© Copyright IBM Corporation 2013

Figure 16-29. Installation problem determination

WA5913.0

Notes:

Before trying to determine the cause of an installation problem, make sure that you read and meet the hardware and software prerequisites.

The latest information for version 8.5.5 is available at:

<http://www.ibm.com/support/docview.wss?uid=swg27038218>

Installing WebSphere Application Server

- A successful installation of WebSphere Application Server V8 is a two-part process
- First, use the Installation Manager 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
 - Management profiles: Deployment Manager, Job Manager, Admin Agent
 - Application Server profile
 - Custom profile

© Copyright IBM Corporation 2013

Figure 16-30. Installing WebSphere Application Server

WA5913.0

Notes:

You must install WebSphere Application Server and then create a profile. Profile creation is no longer part of initial installation and must be done after the installation finishes.



What can go wrong

- Installation Manager fails to install
- When using Installation Manager any of the processes can fail:
 - Installation
 - Update
 - Modify
 - Uninstall
- Profile creation fails
- The installation verification test (IVT) fails:
 - Fails to start the server
 - Fails to validate one or more components

© Copyright IBM Corporation 2013

Figure 16-31. What can go wrong

WA5913.0

Notes:

Depending on the type of problem you have, whether it is with your application, your server, or your tools, you might receive a message that indicates that something is wrong. Always record the error message that you see. As simple as it sounds, error messages often contain codes that make more sense as you investigate your problem further. You might also receive multiple error messages that look similar, but have subtle differences. By recording the details of each message, you can learn more about the problem.

Installation Manager install and startup issues

- Starting with Installation Manager 1.6.2 for AIX, changes to the Eclipse window system require the `gtk` libraries to be installed to support the GUI
- On Linux, ensure both 32-bit and 64-bit versions of required RPMs are installed
- Review the Preparing Operating System section in the information center for more details
- Always use latest version of Installation Manager
- During installation, if the error `JVM terminated: Exit code=13` is encountered:
 - Ensure the `unzip` command is used to extract the downloaded file
 - Do not use any other extraction command other than `unzip`
 - Do not extract on another operating system and then FTP to a UNIX system

© Copyright IBM Corporation 2013

Figure 16-32. Installation Manager Install and startup Issues

WA5913.0

Notes:

This slide has common problems that can happen when trying to install Installation Manager itself.

See the support document, *Required gtk libraries for Installation Manager on AIX*:

<http://www.ibm.com/support/docview.wss?uid=swg21631478>

For common Issues starting or running Installation Manager on UNIX, see the support document, *Issues starting or running Installation Manager*:

<http://www.ibm.com/support/docview.wss?rs=0&uid=swg21438776>

Issue: Installation Manager was installed to default location in silent mode (1 of 2)

- Locate and open the response file `install.xml` in the Installation Manager installation kit
- You must use the profile section to specify the installation location
 - Restriction: The installation location must be a directory that is named `eclipse`
- This example shows the `install.xml` file before the changes:

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean='true' temporary='true'>
    <server>
        <repository location='.'/>
    </server>

    <install>
        <offering features='agent_core,agent_jre'
            id='com.ibm.cic.agent' version='1.6.2000.20130301_2248' />
    </install>
</agent-input>
```

© Copyright IBM Corporation 2013

Figure 16-33. Issue: Installation Manager installed to default location in silent mode (1 of 2)

WA5913.0

Notes:

When installing Installation Manager with silent mode, it installs to default location. To install Installation Manager to a different location, add the profile option to the `install.xml` file or use the command-line option.

Issue: Installation Manager was installed to default location in silent mode (2 of 2)

- This example shows the `install.xml` after the changes:

```
<?xml version="1.0" encoding="UTF-8"?>
<agent-input clean='true' temporary='true'>

    <profile kind='self' installationLocation='/opt/IBM/InstallationManager/eclipse'
        id='IBM Installation Manager'
    </profile>

    <server>
        <repository location='.'/>
    </server>

    <install>
        <offering features='agent_core,agent_jre'
            id='com.ibm.cic.agent' version='1.6.2000.20130301_2248'/>
    </install>
</agent-input>
```

© Copyright IBM Corporation 2013

Figure 16-34. Issue: Installation Manager installed to default location in silent mode (1 of 2)

WA5913.0

Notes:

To install Installation Manager to a different location, add the profile option to the `install.xml` file. This example shows how the `install.xml` file is modified to change the installation location.

16.4.Failure of the installation, update, and uninstall processes

Failure of the install, update, and uninstall processes



© Copyright IBM Corporation 2013

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

8.0

Figure 16-35. Failure of the install, update, and uninstall processes

WA5913.0

Notes:

Issues with installing offerings

- In graphical mode, the error message is logged in graphical panel
 - Review the error message for more details and check logs to review the issue
- Make sure that enough disk space is available for install or update operations
- Never move, edit, or delete any files under Data Agent or IMShared directories
- Review the log file that is created with `-log` option for errors, warnings, or success messages
- Required operating system prerequisites must be completed according to the information center instructions
- `ulimit` for open files on UNIX systems must be increased according to the tuning options that are provided in the information center for each operating system

© Copyright IBM Corporation 2013

Figure 16-36. Issues with installing offerings

WA5913.0

Notes:

These tips are common advice to take care when installing offerings. Offerings mean any product that can be installed by using Installation Manager, such as WebSphere, IBM HTTP Server, AppClient, and the web server plug-in.

Issue: Installation Manager hangs when repositories on NFS mount

- Symptoms:
 - Installation Manager is hung or slow during the install or update operation when repositories are on NFS mount
- Solution:
 - Go to the `IM_DIRECTORY/eclipse/configuration` directory
 - Edit the `config.ini` file with a text editor
 - Add the following line to the bottom of the file: `cic.repo.locking=false` to disable file-locking on repository files
 - This setting can improve performance reading data on NFS
- If the problem still persists, contact IBM Support

© Copyright IBM Corporation 2013

Figure 16-37. Issue: Installation Manager hangs during when repositories on NFS mount

WA5913.0

Notes:

During WebSphere Application Server installation, if the repository is on an NFS mount, hanging or slow response is common. This slide addresses the issue.

Issue: IMShared is using too much disk space

- The shared resources directory can use too much disk space
 - To save on space, you can select not to save files for rollback
- Graphical mode
 - Select **File > Preferences > Files for Rollback**
 - Clear “**Save files for RollBack**”
- Silent mode
 - In response file under preferences, set the following option to false


```
com.ibm.cic.common.core.preferences.preserveDownloadedArtifacts = false
```
- If you use this approach, then repositories for fix packs to which a rollback is done must be added to Installation Manager preferences
- Note: Do not delete or move files manually from IMShared

© Copyright IBM Corporation 2013

Figure 16-38. Issue: IMShared is using too much disk space

WA5913.0

Notes:

The size of IMShared increases for every update since files are stored in this directory. If this option is set, then during rollback, the repository for the fix pack to which you are doing rollback should be added back.

Even after setting this option, Installation Manager requires some files, which should not be deleted from IMShared.

- Command line: imcl -c > select P > then 3



Issue: Installation Manager is not able to find an installed offering

- Make sure that you are running Installation Manager with the correct user ID
 - The user who did the initial installation
- If the `-dL` or `dataLocation` option was used during initial install, make sure to specify `-dL` or `dataLocation` option when starting Installation Manager

© Copyright IBM Corporation 2013

Figure 16-39. Issue: Installation Manager is not able to find an installed offering

WA5913.0

Notes:

Installation Manager should always be started with same user that was initially used unless using group mode. Using a different user results in Installation Manager not being able to detect the installations.

If `-dL` was already used a different Data Agent directory for WebSphere Application Server installation, then `-dL` should be used again and pointed to that directory.



Using the Installation Manager reinstall option

- Installation Manager might become unresponsive because an update of the Installation Manager failed, or the program files became corrupted
 - The `-reinstallIM` option can be used to recover from this situation
 - This option is available only with 1.6.0 and higher
- For command usage, see the information center topic, reinstalling Installation Manager in wizard mode

© Copyright IBM Corporation 2013

Figure 16-40. Using the Installation Manager reinstall option

WA5913.0

Notes:

If the Installation Manager runtime gets corrupted, then the `-reinstallIM` option can be used to resolve it. You can reinstall same or higher version and must use Installation Manager 1.6.0 or higher for this option.

See the topic “Reinstalling Installation Manager in wizard mode” at:

http://pic.dhe.ibm.com/infocenter/install/v1r6/index.jsp?topic=%2Fcom.ibm.cic.agent.ui.doc%2Ftopics%2Ft_manually_installing.html

Recovering from a failed or hung installation

- Depending on the state of your system when an installation fails or hangs, you might uninstall WebSphere Application Server before you try the process again
- Installation Manager must be used to uninstall the offering
 - During uninstall, some files are left under `WAS_HOME` preventing you from reinstalling into the same directory
- If you are using a web repository and the installation fails during download process, make sure that there is no firewall or network issues
 - Select not to keep fetched files during download operation
- Installation Manager cannot be uninstalled while any offering that was installed with it exists
- Refer to the information center for documented procedures

© Copyright IBM Corporation 2013

Figure 16-41. Recovering from a failed or hung installation

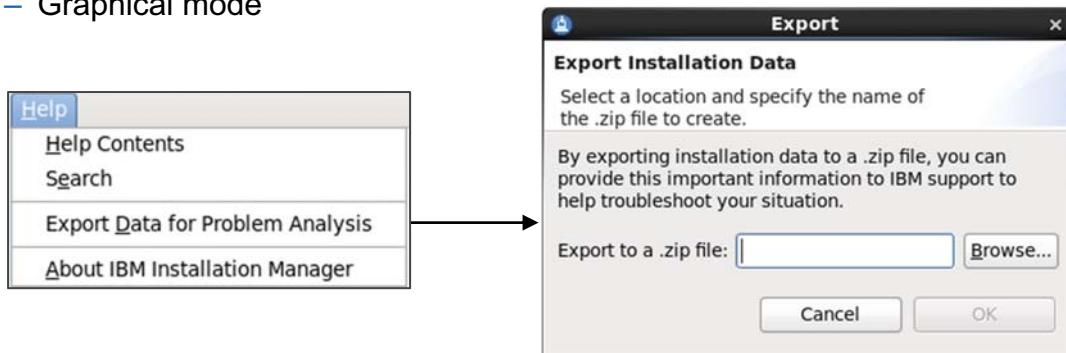
WA5913.0

Notes:

During a failed installation, depending on the state of the installation, you might need to uninstall the offerings. Installation Manager should be used to do uninstall operation. Installation Manager itself cannot be uninstall or remove if there are other products on same machine that use the same Installation Manager.

Collecting data for install, update, and uninstall issues

- Export data
 - Command line: `imcl exportInstallData IMDATA.zip`
 - Graphical mode



- Log created by using `-log` option during the operation
- Logs directory from `HOME` directory of product that was installed, updated, or uninstalled
- Response file or command that is used to install
- IBM Support Assistant collector can be used to collect data

© Copyright IBM Corporation 2013

Figure 16-42. Collecting data for install, update, and uninstall issues

WA5913.0

Notes:

This slide shows the data that support requires and how to collect it in graphical mode and silent mode.

16.5. Profile creation problems

Profile creation problems



© Copyright IBM Corporation 2013

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

8.0

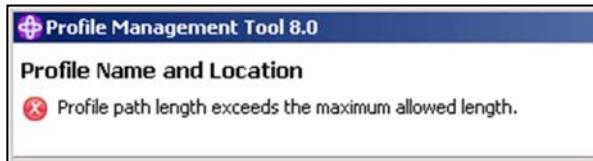
Figure 16-43. Profile creation problems

WA5913.0

Notes:

Profile creation failure

- Problems with profile creation might be due to:
 - Long directory paths (Windows only)



- File permissions

```
<record>
...
<message>com.ibm.wsspi.profile.WSProfileException:
java.io.FileNotFoundException:
C:\Program Files\IBM\WebSphere\AppServer\profiles\AppSrv01
\configcells\4235345Cell01\nodes\4235345Node01\serverindex.xml (Access is denied.)
...
</record>
```

- In graphical mode, after the core product files are installed, Installation Manager gives you the option to start the Profile Management Tool
 - Profiles can also be created at any time after installation

© Copyright IBM Corporation 2013

Figure 16-44. Profile creation failure

WA5913.0

Notes:

Profile creation common reasons to failures are listed here.

Here are some examples of file permissions:

- During profile creation as a nonroot user, if you click **Next** on the Environment Selection dialog, it does not appear to do anything. You are unable to continue creation of a profile, regardless of what profile type was chosen.
- This issue is a permissions issue that the automatic port detection routine caused. If any profile exists and a different user created it, the port detection routine tries to open the `serverindex.xml` file in each profile and fails.

Verify that profile creation succeeded

- Determine whether the errors occurred in the profile creation process
- If the file copy and configuration processes succeed, any error messages further in the log indicate problems in profile creation or later steps, including:
 - Sample application deployment
 - Administrative console application deployment
- Look for the following entry in the file:
 - <WAS_install_root>/logs/manageprofiles/log/<profile_name>_create.log

```
<record>
...
<class>com.ibm.wsspi.profile.WSProfileCLI</class>
<method>invokeWSProfile</method>
<thread>0</thread>
<message>Returning with return code: INSTCONFSUCCESS</message>
</record>
```

- If you do not see this message, there are problems with profile creation

© Copyright IBM Corporation 2013

Figure 16-45. Verify that profile creation succeeded

WA5913.0

Notes:

Look for the following entry in

<WAS_install_root>/manageprofiles/log/<profile_name>_create.log:

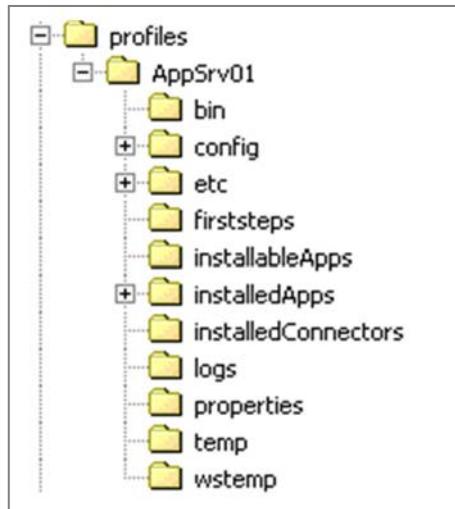
```
<record>
...
<class>com.ibm.wsspi.profile.WSProfileCLI</class>
<method>invokeWSProfile</method>
<thread>0</thread>
<message>Returning with return code: INSTCONFSUCCESS</message>
</record>
```

If you do not see this message, then there are problems with profile creation.

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
 - management
 - managed



© Copyright IBM Corporation 2013

Figure 16-46. Profile creation steps

WA5913.0

Notes:

Profile templates consist of XML configuration files, properties files, Ant tasks, and more.

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 is completed and the creation of a default profile starts
 - Also created when the `manageprofiles` command is run
 - Traces all the events that occur during profile creation
- The log file is an XML log file and can be viewed with:
 - A web browser
 - WordPad in Windows
 - Log Analyzer (IBM Support Assistant and Rational Application Developer)
- The entries in this log consist of `<record>... </record>` stanzas

© Copyright IBM Corporation 2013

Figure 16-47. Data to collect if profile creation fails (1 of 3)

WA5913.0

Notes:

A sample log entry that indicates an error would look similar to the following entry:

```

<record>
<date>2013-07-19T14:18:53</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>

```

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

© Copyright IBM Corporation 2013

Figure 16-48. Data to collect if profile creation fails (2 of 3)

WA5913.0

Notes:

Each log might or might not exist depending on the type of profile that is created.

- 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 that records 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 that is created for server1
- `addNode.log`: Log for federating a node to a cell (custom profile)

Data to collect if profile creation fails (3 of 3)

- Each profile has a `portdef.props` file that is at
 - `<profile_root>/properties/portdef.props`
- Examine this file if you have a port conflict:

```
#Updated portdef.props
#Mon Jun 03 18:55:36 EDT 2013
IPC_CONNECTOR_ADDRESS=9632
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS=9407
XDAGENT_PORT=7066
OVERLAY_UDP_LISTENER_ADDRESS=11039
WC_adminhost=9064
DataPowerMgr_inbound_secure=5555
DCS_UNICAST_ADDRESS=9352
BOOTSTRAP_ADDRESS=9809
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS=9408
SOAP_CONNECTOR_ADDRESS=8879
CELL_DISCOVERY_ADDRESS=7278
ORB_LISTENER_ADDRESS=9102
STATUS_LISTENER_ADDRESS=9442
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS=9409
OVERLAY_TCP_LISTENER_ADDRESS=11040
WC_adminhost_secure=9045
```

© Copyright IBM Corporation 2013

Figure 16-49. Data to collect if profile creation fails (3 of 3)

WA5913.0

Notes:

For troubleshooting port conflicts, you can use the Port Scanner tool in the IBM Support Assistant. You can also use the OS command `netstat -anp` (Linux) `netstat -anb` (Windows) to examine used ports.



What to look for if profile creation fails

- 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 that precede them

© Copyright IBM Corporation 2013

Figure 16-50. What to look for if profile creation fails

WA5913.0

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 with creating a custom node, then look at the log file <WAS_install_root>/profiles/<profile>/logs/addNode.log for any errors.

The following are common problems with profile creation.

File path length errors

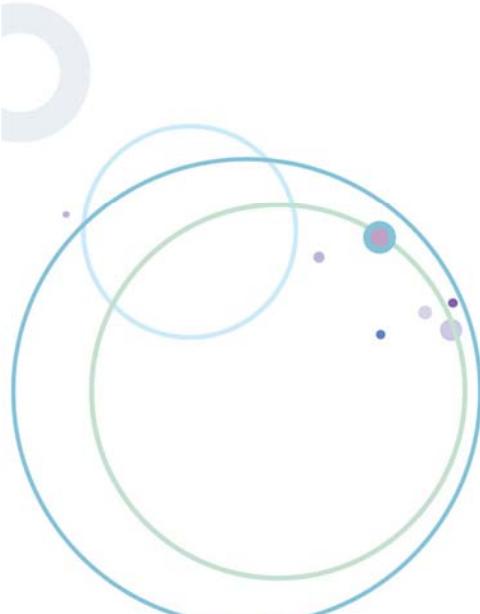
Windows has a length restriction of 258 characters for a command. A problem can occur that prevents the successful creation of a profile when a path is too long. The maximum length for the <WAS_install_root> directory is 60 characters. The maximum length for the profiles installation root directory is 80 characters.

Your log files might have errors similar to "Input line is too long." If so, the length of the file path and node name in the command string 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, and this time use shorter directory paths and node names. If you are still in the installation process, consider reinstalling with a shorter path for the installation root.

16.6.Centralized Installation Manager (CIM)

Centralized Installation Manager (CIM)



© Copyright IBM Corporation 2013

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

8.0

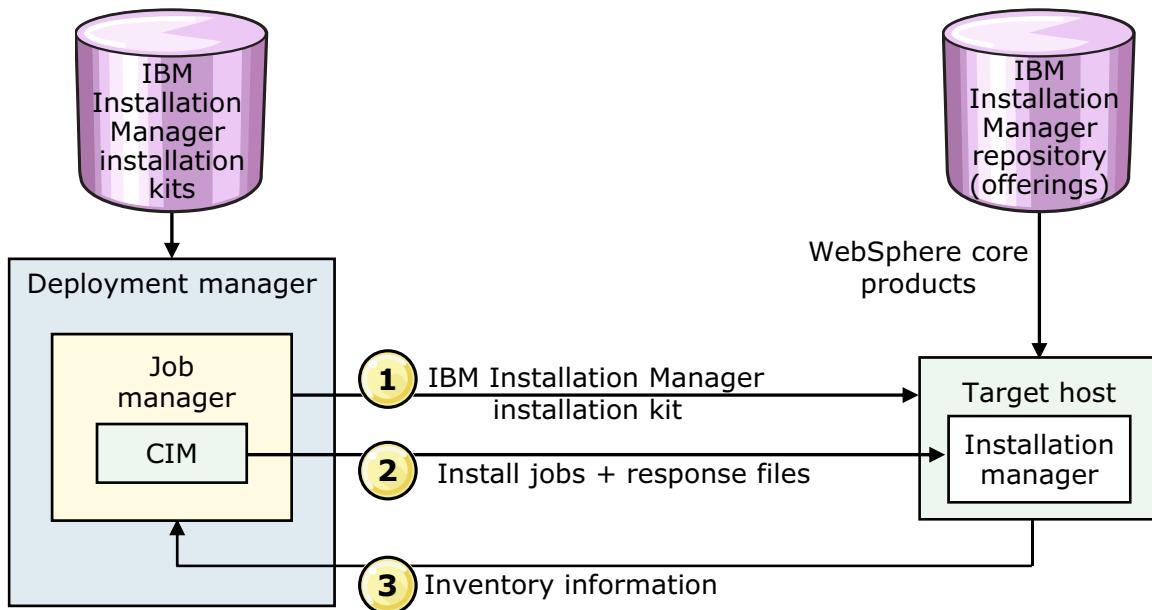
Figure 16-51. Centralized Installation Manager (CIM)

WA5913.0

Notes:

Centralized Installation Manager (CIM) in the job manager

- Architecture of job manager, target hosts, and IBM Installation Manager repository



© Copyright IBM Corporation 2013

Figure 16-52. Centralized Installation Manager (CIM) in the job manager

WA5913.0

Notes:

CIM functionality is now a series of jobs available to the deployment manager's job manager. The job manager can store IBM Installation Manager installation kits for different operating systems in a repository on the local file system. A job can be submitted that "pushes" the required IBM Installation Manager kit to the target node and performs the remote installation with nothing more than target identification and the required authorizations. No agents are required on the target host. Installation jobs for WebSphere Application Server can be configured along with an appropriate response file for the remote target host. When installed on the target host, the Installation Manager can pull the WebSphere code from an IBM Installation Manager repository.

You can submit the **Inventory job** to refresh data on job types and on managed resources of the job manager. Resources include applications and servers of each target. If you installed a product that adds job types on a managed target, run the Inventory job to refresh data on job types and target resources. You can view the refreshed data in the job manager console.

If you plan to use CIM, it is good to set up Enterprise repositories and point response files to it.

Centralized installation manager problems

- For problems that occur when you use CIM to install products or updates, use the MustGather documentation to collect diagnostic data
- Enable the following trace string on the deployment manager
 - CIM commands and runtime: `cimgr=all`
 - CIM WebUI: `com.ibm.ws.console.cim.*=all`
- Possible problems
 - CIM configuration
 - Target host connectivity and authentication
 - Installation Manager repository access

© Copyright IBM Corporation 2013

Figure 16-53. Centralized installation manager problems

WA5913.0

Notes:

See “MustGather: Centralized Installation Manager problems for WebSphere Application Server 7.0, V8.0, and V8.5”: <http://www.ibm.com/support/docview.wss?uid=swg21368123>

16.7.Update and maintenance

Update and maintenance



© Copyright IBM Corporation 2013

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

8.0

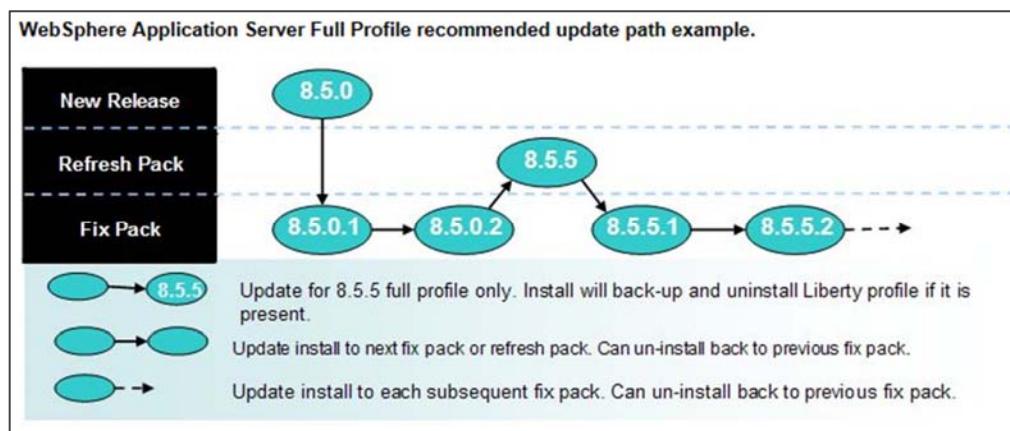
Figure 16-54. Update and maintenance

WA5913.0

Notes:

Update strategy for WebSphere Application Server V8.5.5

- Version 8.5.5 is available as a direct download
 - Download WebSphere Application Server Network Deployment V8.5.5 from Passport Advantage Online
- Existing WebSphere Application Server 8.5.0 installs can be updated to 8.5.5



© Copyright IBM Corporation 2013

Figure 16-55. Update Strategy for WebSphere Application Server V8.5.5

WA5913.0

Notes:

See the support topic, **8.5.5: WebSphere Application Server V8.5.5.0:**

<http://www.ibm.com/support/docview.wss?uid=swg24034969>

See the support topic: **How to download WebSphere Application Server Network Deployment V8.5.5 from Passport Advantage Online:**

<http://www.ibm.com/support/docview.wss?uid=swg27038624>

V8.5.5 is available on Passport Advantage to be installed directly to 8.5.5 for new installations. The existing 8.5.0 installation fix pack can be installed to upgrade 8.5.0 to 8.5.5:

<http://www.ibm.com/support/docview.wss?uid=swg27036014>



Verify current version levels and applied updates

- Use the following tools from <WAS_ROOT>/bin:
 - Show build version, maintenance packages that are installed or uninstalled:

```
versionInfo.bat/sh [ -long, -maintenancePackages,-maintenancePackageDetail, ]
```

- Pinpoint specific interim fix, fix pack, or refresh pack information:

```
historyInfo.bat/sh [ -maintenancePackageID ID_of_maintenance_package ]
```

- From <IM_HOME>/eclipse/tools

```
imcl.bat/sh listInstalledPackages -long
```

- From the administrative console,

- Select **Application servers server_name > Runtime tab > Product Information**

© Copyright IBM Corporation 2013

Figure 16-56. Verify current version levels and applied updates

WA5913.0

Notes:

VersionInfo on WebSphere Application Server can be done from WAS_ROOT/bin.

You can also run the imcl listInstalledPackages command to confirm the version of products installed. The version should always match with WebSphere Application Server versionInfo.

Fix pack download and update

- Use latest version of Installation Manager
- Download fix pack for same offering and extract them to a common directory
 - Fixpack belonging to same offering should be extracted to same common directory
 - IBM must split the fix packs to multiple compressed files due to size of fix packs
- If applying a fix pack to a system where Installation Manager has access to internet, then web repositories can be used
- WebSphere Application Server and JDK fix packs are bundled together, which means, when you apply a fix pack, both WebSphere and JDK get upgraded

© Copyright IBM Corporation 2013

Figure 16-57. Fix pack download and update

WA5913.0

Notes:

Always use the latest Installation Manager version. Download and extract all parts for a fix pack for the same offering to a common directory. If the machine has access to the Internet, then web repositories can be used.

In version 8, WebSphere Application Server and SDK fix packs are bundled together. There is no separate download for 32-bit or 64-bit.

Installing maintenance packages (1 of 2)

- Download and update the latest Installation Manager
- 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 8.5.5.1, regression tested
 - **Refresh pack:** Cumulative package that includes minor new features and fixes
- Update repository location under Installation Manager to do updates
- Run the `backupConfig` command and backup directories that the Installation Manager uses

© Copyright IBM Corporation 2013

Figure 16-58. Installing maintenance packages (1 of 2)

WA5913.0

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 whether an interim fix must be reinstalled.

Refresh packs uninstall all fix packs and interim fixes that are previously applied to the release. So, it is necessary to check the list of delivered fixes to determine whether an interim fix must be reinstalled.



Installing maintenance packages (2 of 2)

- Run the `versionInfo` command
- Always use the Installation Manager to install the interim fix, fix pack, or refresh pack
- Run the `versionInfo` command again to confirm the update
- Verify by using the `imcl listInstalledPackages` command
- Check Installation Manager logs and output on prompt for any issues
- Check log created with `-log` options for any error or warnings

© Copyright IBM Corporation 2013

Figure 16-59. Installing maintenance packages (2 of 2)

WA5913.0

Notes:

The **versionInfo** tool displays important data about the product and its installed components, such as the build version and build date. History information for installation and removal of maintenance packages also displays in the report. This tool is especially useful when working with support personnel to determine the cause of any problem.



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/support/docview.wss?rs=180&uid=swg27006921>

- Determine what version and fix pack level is installed
 - Run the `versionInfo` command from `<WAS_install_root>/bin`
 - Run `imcl listInstalledPackages` from
`<IM_HOME_ROOT>/eclipse/tools`
- Read the MustGather documentation for installation problems
<http://www.ibm.com/support/docview.wss?uid=swg21497417>

© Copyright IBM Corporation 2013

Figure 16-60. Before contacting IBM support

WA5913.0

Notes:

The link to prerequisites given in the slide describes the minimum product levels that you should install before opening a problem report with the WebSphere Application Server support team.



Backing up Installation Manager

- It is recommended to back up IMShared and data agent at the same date and time
 - Before and after applying a fix pack, fixes, or any installation that is done using the Installation Manager
- It is recommended to back up all three directories
 - Offering
 - IMShared
 - Data Agent
- Restore all directories from the same time frame

© Copyright IBM Corporation 2013

Figure 16-61. Backing up Installation Manager

WA5913.0

Notes:

This recommendation for backup is applicable if you are updating WebSphere Application Server. If updating any additional offering that uses Installation Manager at the same time, follow documentation for the backup of the offering in addition to these steps.

For Installation Manager backup, see the information center topic, *Backing up and restoring Installation Manager*.

http://pic.dhe.ibm.com/infocenter/install/v1r6/index.jsp?topic=%2Fcom.ibm.cic.agent.ui.doc%2Ftopics%2Ft_im_backup.html

If **only** the home directory of an offering is restored after failure during update or installation, then the Installation Manager Data gets out of sync with installed offerings like WebSphere Application Server, IBM HTTP Server, and plug-ins. There is no way to recover it except to reinstall all the products that Installation Manager manages.

Unit summary

Having completed this unit, you should be able to:

- Identify common installation problems
- Locate and examine relevant installation logs
- Recover from a failed installation
- Search for relevant version information and prerequisite levels
- Identify problems that are associated with applying maintenance updates

© Copyright IBM Corporation 2013

Figure 16-62. Unit summary

WA5913.0

Notes:

Checkpoint questions

1. (T/F) The Installation Manager GUI can be used to view all installed packages on a host system.
2. (T/F) As of V8.5, only the `manageprofiles` command can be used to create an application server profile.
3. (T/F) If the Shared Resources Directory is using too much disk space, you can configure the Installation Manager not to save files for rollback.
4. (T/F) The job manager can push an IBM Installation Manager kit to a target host and install it remotely.

© Copyright IBM Corporation 2013

Figure 16-63. Checkpoint questions

WA5913.0

Notes:

Write your answers here:

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



Checkpoint answers

1. True
2. False. You can still use the Profile Management Tool to create an application server profile.
3. True
4. True

© Copyright IBM Corporation 2013

Figure 16-64. Checkpoint answers

WA5913.0

Notes:

Unit 17. Intelligent Management problem determination and problem determination tools

What this unit is about

This unit describes Intelligent Management and how it can be used to troubleshoot common problems.

What you should be able to do

After completing this unit, you should be able to:

- Describe the basic functions of Intelligent Management
- Troubleshoot Intelligent Management routing problems
- Troubleshoot application placement problems
- Use the health management feature of Intelligent Management
- Use other Intelligent Management features to help with problem determination

How you will check your progress

- Checkpoint questions
- Lab exercise

References

WebSphere Application Server Information Center topic,

“Intelligent Management Overview”:

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.wve.doc%2Fae%2Fcwve_xdovrvw.html

Unit objectives

After completing this unit, you should be able to:

- Describe the basic functions of Intelligent Management
- Troubleshoot Intelligent Management routing problems
- Troubleshoot application placement problems
- Use the health management feature of Intelligent Management
- Use other Intelligent Management features to help with problem determination

© Copyright IBM Corporation 2013

Figure 17-1. Unit objectives

WA5913.0

Notes:



Topics

- Intelligent Management overview
- Intelligent Management problem determination
- Intelligent Management problem determination tools

© Copyright IBM Corporation 2013

Figure 17-2. Topics

WA5913.0

Notes:

17.1.Intelligent Management overview

Intelligent Management overview



© Copyright IBM Corporation 2013

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

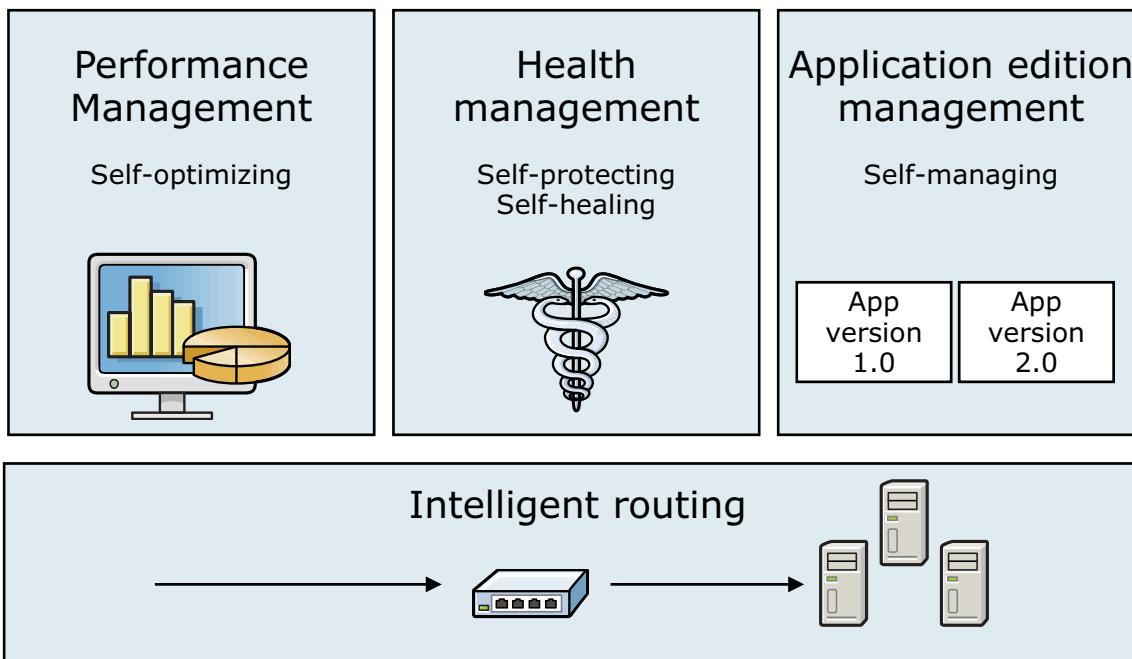
8.0

Figure 17-3. Intelligent Management overview

WA5913.0

Notes:

Intelligent Management



© Copyright IBM Corporation 2013

Figure 17-4. Intelligent Management

WA5913.0

Notes:

There are four main topics on Intelligent Management.

Overview of Performance Management capabilities:

- Associate service policies with your applications and have WebSphere efficiently manage requests to achieve these goals
- Decrease administrative work that is required to monitor and diagnose performance issues
- Minimize the number of JVMs and virtual machines that are running to reduce processing of lightly used or idle JVMs or virtual machines
- Protect your middleware infrastructure against overload

Overview of health management capabilities:

- Automatically detect and handle application health problems without requiring administrator time, expertise, or intervention
- Intelligently handle health issues in a way that maintains continuous availability
- Configure applications differently if you want to

- Approve automatic actions before the actions are taken
- Set policies that are based on both high-level metrics and low-level metrics to detect problems before you notice that something is wrong

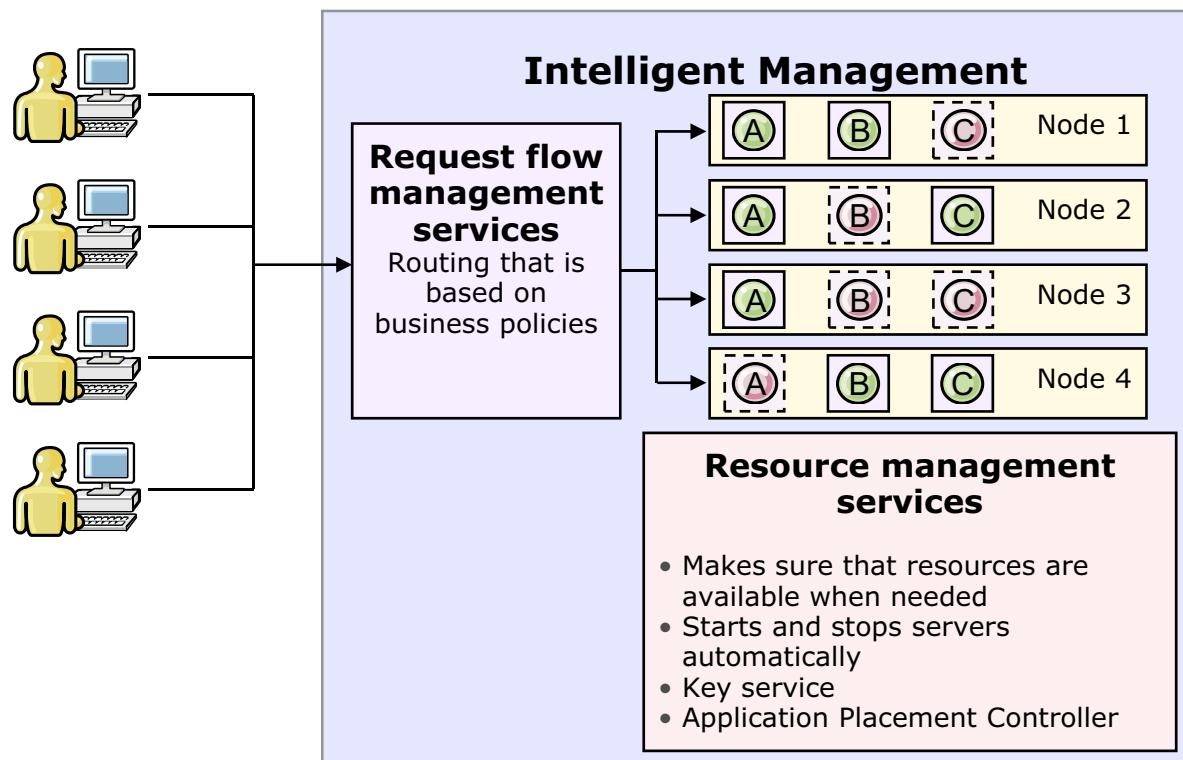
Overview of application edition management capabilities:

- Ability to run multiple versions of your applications concurrently
- Update your applications without incurring downtime
- Verify that a new version of your application works in production before sending real traffic to it
- Reduce your infrastructure costs and decrease potential outages
- Easily update your operating system or WebSphere without incurring downtime

Overview of intelligent routing capabilities:

- A routing tier that is aware of what is happening on the application server tier and reacts to it; business-critical applications are given priority
- Automatic routing without needing to update configuration files
- A highly scalable routing tier
- Ease of management
- Flexible policy-based routing to control if, when, and where requests are routed
- A highly available deployment manager

Performance Management



© Copyright IBM Corporation 2013

Figure 17-5. Performance Management

WA5913.0

Notes:

Performance Management provides two main categories of services. Request Flow Management services intelligently route requests to appropriate resources based on business policies you define. Resource Management services make sure that the right servers are available to meet the demand. One of the key services that Intelligent Management uses to implement Performance Management is the application placement controller.

Health management

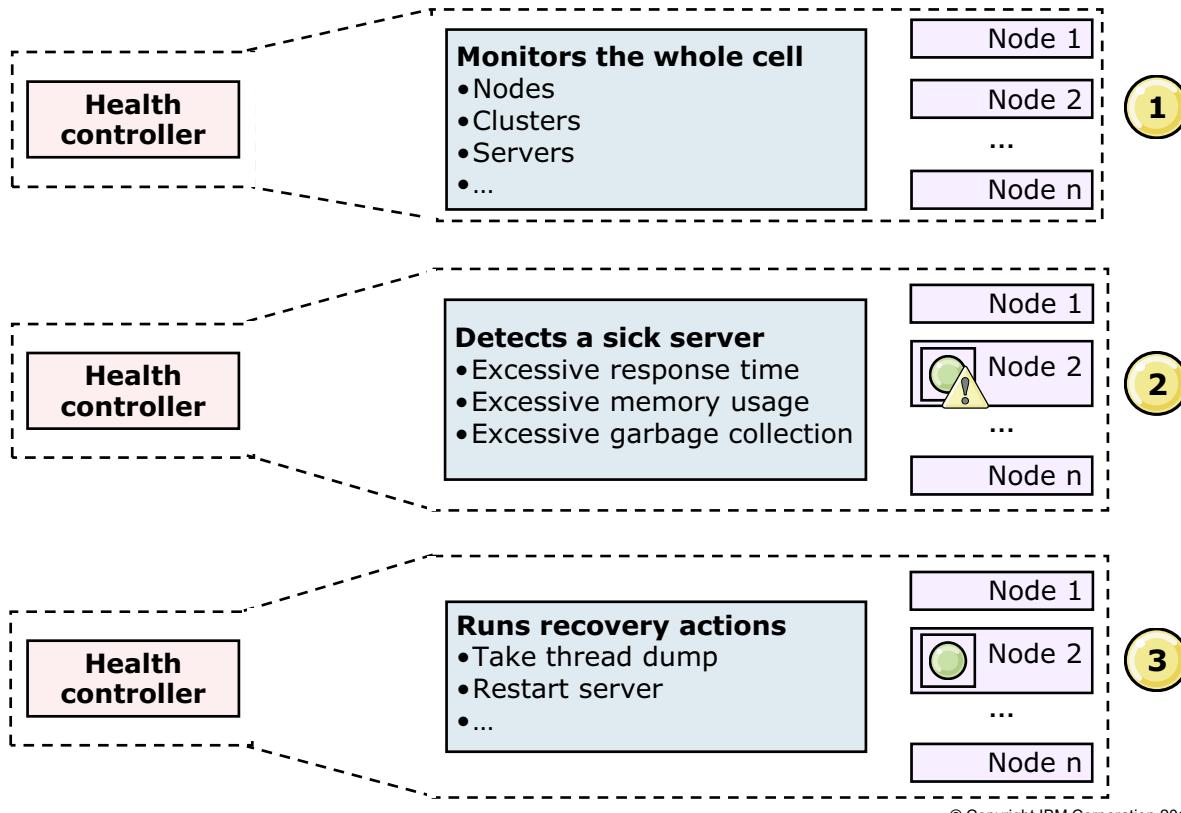


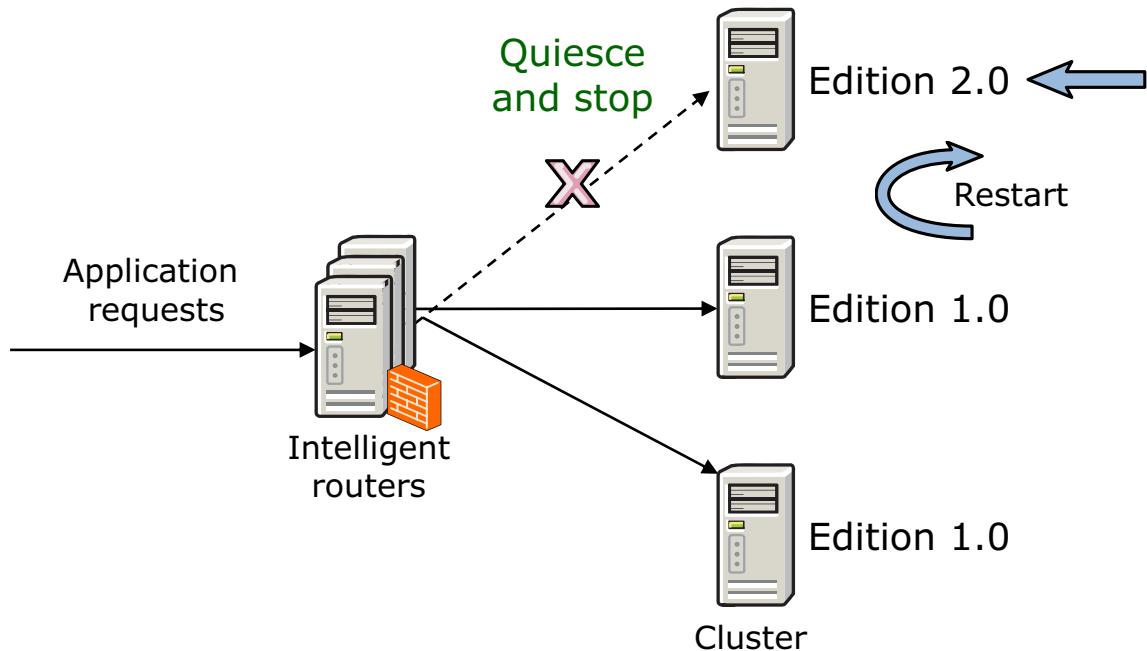
Figure 17-6. Health Management

WA5913.0

Notes:

The Health Management feature of Intelligent Management allows you to monitor your environment and automatically recover when unhealthy servers are found. The autonomic manager responsible for monitoring and recovering servers is called the health controller. You define what unhealthy conditions to watch for, and what actions to take with health policies. The health controller works together with the other autonomic managers to ensure that actions are taken intelligently, so service is not interrupted, and service policies are met.

Application edition management



© Copyright IBM Corporation 2013

Figure 17-7. Application Edition Management

WA5913.0

Notes:

Application edition management enables management of interruption-free production application deployments. Using this feature, you can validate a new edition of an application in your production environment without affecting users, and upgrade your applications without incurring user outages. You can also run multiple editions of a single application concurrently, directing different users to different editions. Intelligent routers maintain not only traditional application state (for example, HTTP session) affinity, but also application version affinity.

This slide shows an example of a group rollout scenario. In the diagram, the cluster consists of three servers. You first must divide the cluster into groups, which tells the application edition manager how many servers to update at the same time. If you do a rollout on a group, the servers in each group are upgraded to the new edition at the same time. Each server in the group is quiesced, stopped, and reset.

As the rollout is run in the diagram on the slide, one server in the cluster is moved from Edition 1.0 to Edition 2.0. During this time, the server does not receive user requests that are directed from the intelligent router, and the server is stopped. All application requests are sent to the servers that are running Edition 1.0. After the server that is running Edition 2.0 is available, the on demand router

directs application requests to that server. Any servers that are still running Edition 1.0 do not serve requests until the edition is updated to Edition 2.0.

Intelligent routing

- A routing tier that is aware of what is happening on the application server tier
 - Knows which cluster members are currently started
 - Knows application server usage, performance, and other statistics
 - Understands service policies
 - Routes work to the application server that can do it best
 - Can route to multiple application editions
 - Provides preference for higher priority requests (some restrictions)
 - Provides processor and memory overload protection (some restrictions)
- Integrates with Performance Management, health management, and application edition management

© Copyright IBM Corporation 2013

Figure 17-8. Intelligent routing

WA5913.0

Notes:

Intelligent routing can improve the quality of service by making the best use of available resources, and ensuring that priority is given to business-critical applications and users. Requests to applications are prioritized and routed based on administrator-defined rules. Intelligent routers retrieve information on the available servers and their capacity to serve requests. Intelligent routers work with Performance Management, health management, and application edition management.

Intelligent Management components

- Dynamic clusters
- Service policies
- Autonomic managers and services
- Intelligent routers (two implementations)
 - On-demand router (ODR)
 - WebSphere plug-in

© Copyright IBM Corporation 2013

Figure 17-9. Intelligent Management components

WA5913.0

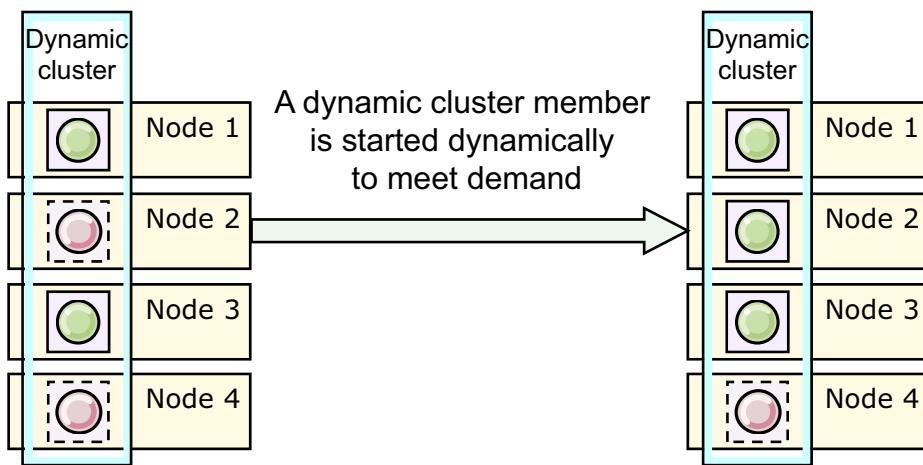
Notes:

Four main components are used to implement Intelligent Management functions.

- Dynamic clusters allow the amount of resources available for application requests to be adjusted dynamically.
- Service policies allow business goals to be used for decision making in how requests are handled.
- Autonomic managers provide information and act to implement Intelligent Management functions.
- Intelligent routers intelligently dispatch incoming requests to the application server tier.

Dynamic clusters

- Similar to a static cluster, but number of active servers can be resized dynamically at run time
 - The number of active instances depends on business policies and current request flow
 - Managed by the application placement controller



© Copyright IBM Corporation 2013

Figure 17-10. Dynamic clusters

WA5913.0

Notes:

Dynamic clusters are built on static clusters, but they have extra behavioral characteristics. The dynamic part of a dynamic cluster is the number of instances that are started. When a dynamic cluster is created, a set of rules is specified that determines which nodes have dynamic cluster member instances that are created on them. All of the members are then statically created on those nodes when the cluster is defined. All of the members exist, but they are started and stopped dynamically.

When a new node is added to the cell, all dynamic clusters with rules that match the characteristics of the new node have a cluster member immediately created. An example of a rule might be: "All nodes in the cell except the node with the Deployment Manager".

Intelligent routing makes sure that requests are routed to the currently started dynamic cluster members.

Service policies

- Associated with a set of requests
 - AppA/*
 - AppB/*
- Define priorities
 - AppA/* requests are more important than AppB/* requests
- Define objectives
 - Need average response time of 1 second or less

Gold service policy
Highest priority
Objective: 1-second response time

Silver service policy
Medium priority
Objective: 2-second response time

Bronze service policy
Lowest priority
Objective: 3-second response time

© Copyright IBM Corporation 2013

Figure 17-11. Service policies

WA5913.0

Notes:

Service policies define how Intelligent Management handles requests. Service policies are associated with a particular protocol; then, within that protocol are regular expressions that specify a specific set of requests. A service policy defines the relative priority of a particular set of requests – from highest to lowest. Intelligent Management can make sure that higher priority requests get more resources than lower priority requests. A service policy also defines an objective, typically a user response time goal. If a set of requests is not meeting its objective, Intelligent Management can allocate more resources to the requests. Service policies can be created without an objective. In this case, resource allocation is based on how long the requests must wait.

Autonomic managers and services

- Application placement controller
 - Decides when and where to start dynamic cluster members
- Autonomic request flow manager
 - Classifies incoming requests and monitors performance
- Dynamic workload management controller
 - Adjusts server weights for dynamic routing
- Health controller
 - Monitors health policies and acts to correct problems
- On-demand configuration service
 - Maintains cell topology information and keeps the other controllers informed about the environment

© Copyright IBM Corporation 2013

Figure 17-12. Autonomic managers and services

WA5913.0

Notes:

Intelligent Management capabilities are delivered in a set of components that are known as autonomic managers. Autonomic managers monitor performance and health statistics through a series of sensors; they optimize system performance and run traffic shaping.

- **The application placement controller**

The application placement controller is responsible for the management of the location of an application. A single application placement controller exists in the cell and is hosted in the deployment manager or in a node agent process. The application placement controller starts and stops application server instances to manage traffic. The application placement controller can dynamically address periods of intense workflow that would otherwise require the manual intervention of a system administrator.

- **The autonomic request flow manager**

The main purpose of the autonomic request flow manager is to determine when requests are sent to the application servers. Intelligent Management calculates the amount of resources that each type of request needs on each node. The ARFM uses information from the nodes, from the ODR, and from PMI statistics.

- **The dynamic workload controller**

The dynamic workload controller dynamically adjusts server weights to even out and minimize response times across the cluster. There is one dynamic workload controller per cluster. The dynamic workload controller maintains a list of active server instances for each dynamic cluster, and assigns each a routing weight according to observed performance trends. Requests are then routed to candidate server instances to balance workloads on the nodes within a cluster.

- **The health controller**

The health controller enforces configured health policies. When policy conditions are violated, the health controller carries out the defined actions for the health policy.

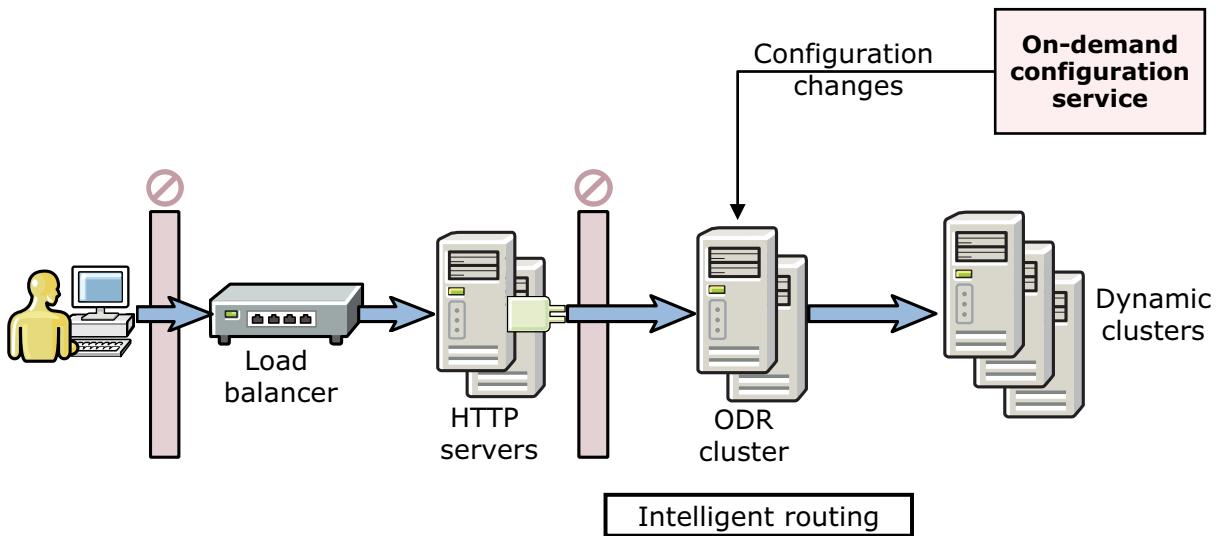
- **The on-demand configuration service**

The on-demand configuration service maintains cell topology information and keeps the other controllers aware of the environment. It tracks updates in cell topology and state, including the following changes:

- Applications that are installed and removed
- Servers started and stopped
- Nodes that added and removed
- Classification updates

The on-demand configuration component allows the intelligent routers to understand the environment.

Intelligent routers: ODR



© Copyright IBM Corporation 2013

Figure 17-13. Intelligent routers: the ODR

WA5913.0

Notes:

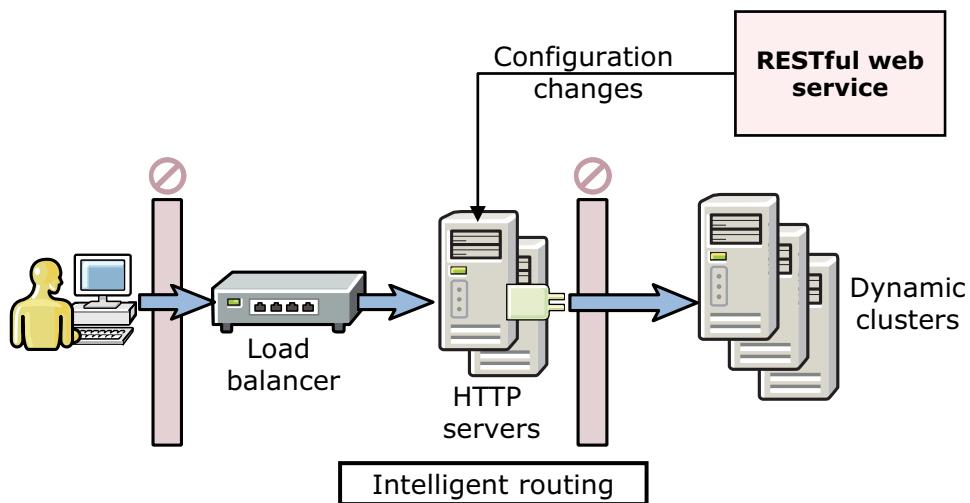
The on demand router (ODR) is a specialized Java based proxy server that classifies incoming requests, and then dispatches the requests across the application server environment. Here the HTTP servers are routing all requests to the on demand routers. The on demand routers handle routing of requests to the application servers. Information about the environment is retrieved from the on-demand configuration service.

The on demand router classifies incoming HTTP and SIP requests and then works with other Intelligent Management “decision makers” to route the workload. The on demand router ensures that the highest priority is given to business-critical applications. Requests are prioritized and routed based on administrator-defined service policies, which are used to specify application response time goals.

The on demand router handles the queuing and dispatching of requests according to operational policy. An on demand router can be defined and started before any service policies are defined. Operational policies can be defined before the appearance of the work to which they apply. However, if policies are not defined, the default policies handle the early work. The on demand router, similar to the web server plug-in for WebSphere Application Server, uses session affinity to route work requests. After a session is established on a server, later work requests for the same

session go to the original server. This configuration maximizes cache usage and reduces queries to resources. The on demand router accepts incoming requests and distributes these requests to the system in an intelligent manner, reflecting configured business goals. This process is dependent on the characterization of requests so that the relative business importance of each request can be compared.

Intelligent routers: WebSphere plug-in



© Copyright IBM Corporation 2013

Figure 17-14. Intelligent routers: WebSphere plug-in

WA5913.0

Notes:

The WebSphere plug-in can also be configured as an intelligent router. The plug-in has many of the same capabilities as the ODR, but does not provide request prioritization or overload protection. The plug-in intelligent router retrieves configuration information from a web service, rather than by using the on-demand configuration service directly.

17.2.Intelligent Management problem determination

Intelligent Management problem determination



© Copyright IBM Corporation 2013

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

8.0

Figure 17-15. Intelligent Management problem determination

WA5913.0

Notes:



Components to watch

- On-demand configuration (ODC)
 - ODC represents the WebSphere configuration
 - Based on process/application events (start/stop) are dynamically updated
 - Configuration changes in Deployment Manager are dynamically updated
 - Every process builds, holds, and contributes ODC data
- Intelligent routers
 - On demand router (ODR)
 - WebSphere plug-in (already covered in the request flow unit)
- Application placement controller (APC)
 - Highly available service
 - Typically runs in a node agent or the dmgr

© Copyright IBM Corporation 2013

Figure 17-16. Components to watch

WA5913.0

Notes:

There are three components to watch when problems occur with Intelligent Management. Many functions of Intelligent Management require accurate information about the current configuration. The on-demand configuration service is responsible for maintaining and distributing the current configuration. Make sure that the ODC is working correctly when debugging Intelligent Management issues.

If you are having routing problems, looking at the intelligent routers is useful.

For problems where servers are not started and stopped as expected, understanding the behavior of the application placement controller is helpful. The application placement controller is the service that decides where and how many servers to start.

Examining on-demand configuration

- ODC can be represented as XML
 - Export to target.xml

```
wsadmin.sh -f manageODC.py getTargetTree <node> <process>
```
 - Information can be large (topology dependent)
- Ensure that configuration repository is synchronized to all processes
 - Force a full sync with all nodes
 - Reinspect new export of target.xml
- Ensure that communication infrastructure is working correctly
 - Rapid server leave/join messages in the logs indicate potential port conflicts
- Collect all logs and the target.xml for support assistance
- Must gather trace can be collected

© Copyright IBM Corporation 2013

Figure 17-17. Examining on-demand configuration

WA5913.0

Notes:

The configuration that is held in a server's memory by the on-demand configuration can be exported to an XML file. A wsadmin script is used to export the configuration to a target.xml file. If the information in the exported target.xml file does not match the actual configuration, make sure that the configuration is synchronized throughout the cell. Also, ensure that there are no network communication problems between the servers in the cell.

On-demand router: Debugging a routing failure

How to use the default ODR access logs

- The `proxy.log` contains return codes from the application server
 - The ODR proxied the request to the application server
 - Errors that are received from the application server are in the `proxy.log`
- The `local.log` contains all return codes from the ODR
 - Errors from the ODR are in the `local.log`
- The `cache.log` shows all cache entries

© Copyright IBM Corporation 2013

Figure 17-18. On-demand router: Debugging a routing failure

WA5913.0

Notes:

The ODR uses three logs. The proxy log contains all of the return codes from requests that were proxied to an application server. The application server produced these return codes. Return codes for errors that the ODR produces are contained in the local log. The cache log shows what requests had responses that are returned from the cache.

ODR: Debugging 404 errors

- Determine whether 404 comes from application server or ODR
 - Look in proxy.log and local.log
- If 404 is in proxy.log, the application server returned the error
 - The most common cause is that a proxy in front of the ODR was not configured as a trusted proxy in the ODR settings panel
 - See: http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=/com.ibm.websphere.wve.doc/ae/twve_ccgodrsen.html
 - Otherwise, check the application server logs
- If the 404 is in local.log, the ODR returned the 404 directly
 - Turn on ODR tracing to resolve the problem
 - See: http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=/com.ibm.websphere.wve.doc/ae/twve_odoebtf.html

© Copyright IBM Corporation 2013

Figure 17-19. ODR: Debugging 404 errors

WA5913.0

Notes:

When a request returns with a 404 error, check the ODR logs to see where the return code was generated. If the 404 is in the proxy log, the application server returned the error. One common cause of this error is the misconfiguration of the plug-in in front of the ODR. If the plug-in configuration did not cause the error, debug the 404 as an application server issue. If the 404 is in the local log, turn on tracing in the ODR and debug the 404 as an ODR issue.

ODR: Debugging 503 errors

- Determine whether 503 comes from application server or ODR
 - Look in `proxy.log` and `local.log`
- If 503 is in `proxy.log`, the application server returned the error
 - Check the application server logs
- If 503 is in `local.log`, the ODR returned the 503 directly
- The ODR routes to an application server only if the following are true
 - Server **state** is “STARTED”
 - Server **weight** is greater than 0
 - Server is not in **maintenanceMode**
 - Server is running the web module of the request
 - Server has **active** HTTP or HTTPS transport definitions
 - Server is not marked as **unreachable**

All of these conditions can be verified in ODC

© Copyright IBM Corporation 2013

Figure 17-20. ODR: Debugging 503 errors

WA5913.0

Notes:

For 503 errors, again, if the error is in the proxy log, debug the error as an application server issue, with techniques that were presented previously.

If the 503 is in the local log, the ODR did not route the request correctly. If the ODR cannot find an application server that can handle the request, the ODR returns a 503. An application server can handle a request only if a set of conditions are true. If no server is found where all of the conditions are true, the ODR cannot route the request. The characteristics of the application servers can be found in the `target.xml` file that the on-demand configuration service generates.

ODR: Understanding the unreachable error

- A server is unreachable when the ODR fails to connect to it
- An OS-level resource can be exhausted on the ODR or the application server
- What resources can be exhausted?
 - File descriptors – too many files open
 - Check the ulimit for the process limit, and the system limit
 - Ephemeral ports – too many sockets that are waiting to close
 - Tune TCP parameters for the operating system
- If unreachable occurs to multiple application servers
 - Most likely due to resource exhaustion on the ODR
- If unreachable occurs to a single application server
 - Most likely due to resource exhaustion on that application server node

© Copyright IBM Corporation 2013

Figure 17-21. ODR: Understanding the unreachable error

WA5913.0

Notes:

A server is marked unreachable if the ODR cannot connect to it. If the server is up, it might mean that an OS-level resource is exhausted on the ODR or the application server. File descriptors and ephemeral ports are two common resources that can be exhausted. If the unreachable error occurs to multiple application servers, it is most likely due to resource exhaustion on the ODR. If the unreachable error occurs to a single application server, it is most likely due to resource exhaustion on the application server node.

Application placement controller issues (1 of 2)

Application placement: service policy goals are not guaranteed

- Intelligent management cannot make a request faster
- Placement does not occur if it does not help with response time
 - Even if a service policy goal is being breached
 - Prevents unrealistic goals from de-stabilizing the environment

Application placement does not occur unless there is capacity

- APC does not over-commit memory on a node
- There might not be enough processor capacity

© Copyright IBM Corporation 2013

Figure 17-22. Application placement controller issues (1 of 2)

WA5913.0

Notes:

You might find issues with the behavior of the application placement controller.

It is important to understand the relationship between the application placement controller and service policy goals. The APC cannot guarantee service policy goals. Intelligent Management cannot make a request faster. If it takes 2 seconds to service a particular request in a normal environment, adding Intelligent Management does not magically allow that request to be served in 1 second. If requests are slowed down because they are waiting for resources, Intelligent Management can help with that problem. If adding more resources cannot improve response time, more cluster members are not started, even if service policy goals are not being met. This behavior prevents unrealistic goals from adding instability into the environment.

Also, more cluster members are not started if the capacity does not exist to support the server instances. The APC understands the memory and processor capacity of each node, and does not start servers that over-commit the capacity.

Application placement controller issues (2 of 2)

- Server start times must be configured appropriately
 - Part of APC configuration is the time to wait for a server to start
 - If the server does not start before the timeout is reached, the server is marked as failed-start
 - If start time is set too low, the APC might begin to start more dynamic cluster members
 - Configure the longest amount of time it takes to start any application server under load
- Dynamic clusters should not be managed manually
 - Application placement controller might undo the operation

© Copyright IBM Corporation 2013

Figure 17-23. Application placement controller issues (2 of 2)

WA5913.0

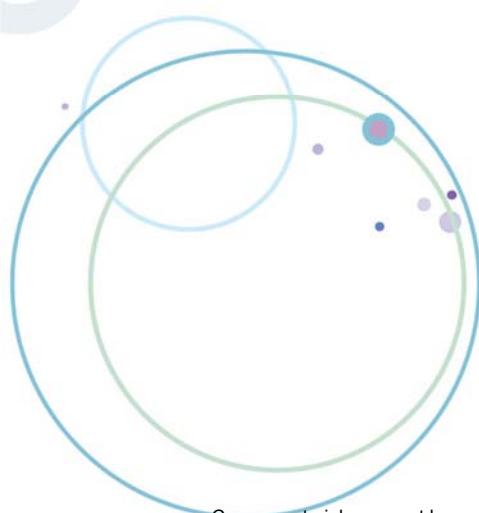
Notes:

Server start times must be configured to inform the APC about how long to wait for a server to start before considering it a failed server. If the start time is set too low, the APC might try to start more cluster members. When a slow server does start, it might cause problems; for example, by using memory that other servers need. Configure the server start time for the APC to the longest time it takes any server to start when the system is under load.

Dynamic clusters should not be managed manually. The APC might reverse any start or stop action.

17.3.Intelligent Management problem determination tools

Intelligent Management problem determination tools



© Copyright IBM Corporation 2013

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

8.0

Figure 17-24. Intelligent Management problem determination tools

WA5913.0

Notes:

Problem determination tools

- Centralized logging
- Request based tracing
- Health management

© Copyright IBM Corporation 2013

Figure 17-25. Problem determination tools

WA5913.0

Notes:

Three problem determination tools added with Intelligent Management are centralized logging, request based tracing, and health management.

Centralized logging: Simplifying log collection

New script: `mustgather.py`

- Easily set trace strings that are based on problem type
 - Trace strings that are used differ based on problem type
 - Servers where trace strings are set differ based on problem type
- Easily return trace strings to default values throughout the cell
- Gather MustGather files centrally upon request
 - Does **not** stream logs to a central location at run time
- Example: Collect MustGather information for 503 error codes
 - `wsadmin.sh -f mustgather.py enable 503`
 - ## Reproduce the problem
 - `wsadmin.sh -f mustgather.py disable 503`
 - `wsadmin.sh -f mustgather.py collect 503 /tmp/503_logs.zip`

© Copyright IBM Corporation 2013

Figure 17-26. Centralized logging: Simplifying log collection

WA5913.0

Notes:

The centralized logging feature allows trace strings to be set for multiple servers from a single script. The feature also allows the generated logs to be gathered to a central location.

The script for setting trace strings and collecting log information is called `mustgather.py`. Depending on the type of problem, different trace strings are needed. Also, the servers that must be traced might be different. Based on a specified problem type, the `mustgather.py` script sets the appropriate trace strings on the appropriate servers. After the trace strings are set, you either re-create the problem, or wait for the problem to occur. You then use `mustgather.py` to disable the trace strings, and gather the MustGather artifacts to a central location. The collected artifacts can then be analyzed or sent to IBM support.

It is important to understand that `mustgather.py` does not cause any log messages or other information to be streamed to a central location. It is used after the fact to collect the artifacts to a central location.

Centralized logging: Simplifying log collection

- Supported MustGather types
 - 404, 503, 504 – HTTP response codes
 - agent – Middleware agent
 - apc – Application Placement Controller
 - appedition – Application Edition Manager
 - arfm – Autonomic Request Flow Manager
 - dc – Dynamic clusters
 - hadmgr – High Availability Deployment Manager
 - hmm – Health monitoring
 - odr – General issues on the on demand router
 - operations – Visualization issues with Component Stability and Operations tabs
 - reports – Visualization issues with Report tab
 - reportsPerf – Visualization issues with performance data shown on Report tab
 - repository – Extended Repository Service
 - request – Request based tracing
 - sip – SIP request routing

© Copyright IBM Corporation 2013

Figure 17-27. Centralized logging: simplifying log collection

WA5913.0

Notes:

Here is a reference slide to show the different problem types that the `mustgather.py` script recognizes.

Request based tracing

New script: `setReqBasedTracing.py`

- Dynamic per-request trace enablement
 - Minimizes processor usage
 - Minimizes volume of data collected
- Enables a trace string in an ODR and single application server during a particular request
 - Based on request-matching rules
- Clears trace (both app server and ODR) when the response is sent back to client, if no other matching requests are being processed
 - Trace is enabled for **ALL** requests during this period, including requests not matching the trace rule
 - **ALL** trace is disabled after response is completed (reverted to *=info)

© Copyright IBM Corporation 2013

Figure 17-28. Request based tracing

WA5913.0

Notes:

A new script that is called `setReqBasedTracing.py` provides the capability to enable tracing based on the arrival of a particular request. Tracing is turned on in the ODR, and in the application server where the request is routed. The script sets the trace strings on the ODR and application server, so trace messages for all requests during this time are written. When the request completes, the trace string is set to `*=info`; it is not reset to whatever it was previously.

Request based tracing

- Useful for enabling trace when
 - Problem traffic is a small subset of overall traffic
 - Problem traffic can be identified with a rule
- Example:
 - ```
./wsadmin.sh -f setReqBasedTracing.py enableReqBasedTracing
 -ruleExpression:"uri like \'%/StockTrade/%\'"
 -odrTraceSpec:com.ibm.ws.proxy.*=all
 -appServerTraceSpec:com.ibm.ws.webcontainer.*=all
 -RuleID:StockTradeRule
 ## Wait for problem to occur
 ./wsadmin.sh -f setReqBasedTracing.py
 disableReqBasedTracing -RuleIDs:StockTradeRule
```
- Rule expressions same as for service or routing policies
  - Header value, query parameter, URI, and others
- Typically, trace strings would be more complex than example

© Copyright IBM Corporation 2013

Figure 17-29. Request based tracing

WA5913.0

### Notes:

Request-based tracing is useful if the problem traffic is only a small subset of the overall traffic, and a rule can be specified to recognize the problem traffic. Here is an example that uses the `setReqBasedTracing.py` script.

The rule expressions are the same as for work class rule expressions. Rule IDs are associated with a rule expression and a set of trace strings. When the expression is true for a request, runtime trace strings are set on the ODR where the request arrived, and on the application server where the request is sent.

After trace information is collected, the `setReqBasedTracing` script is used to disable a particular ruleID.

## Health management overview

- WebSphere Application Server Intelligent Management monitors your environment for various health conditions
- The health controller monitors health conditions
- When a condition is violated, the health controller runs a sequence of actions
- Health policies define:
  - Conditions to monitor
  - Actions to run
- Restart actions are done intelligently to prevent outages and service policy violations

© Copyright IBM Corporation 2013

Figure 17-30. Health Management overview

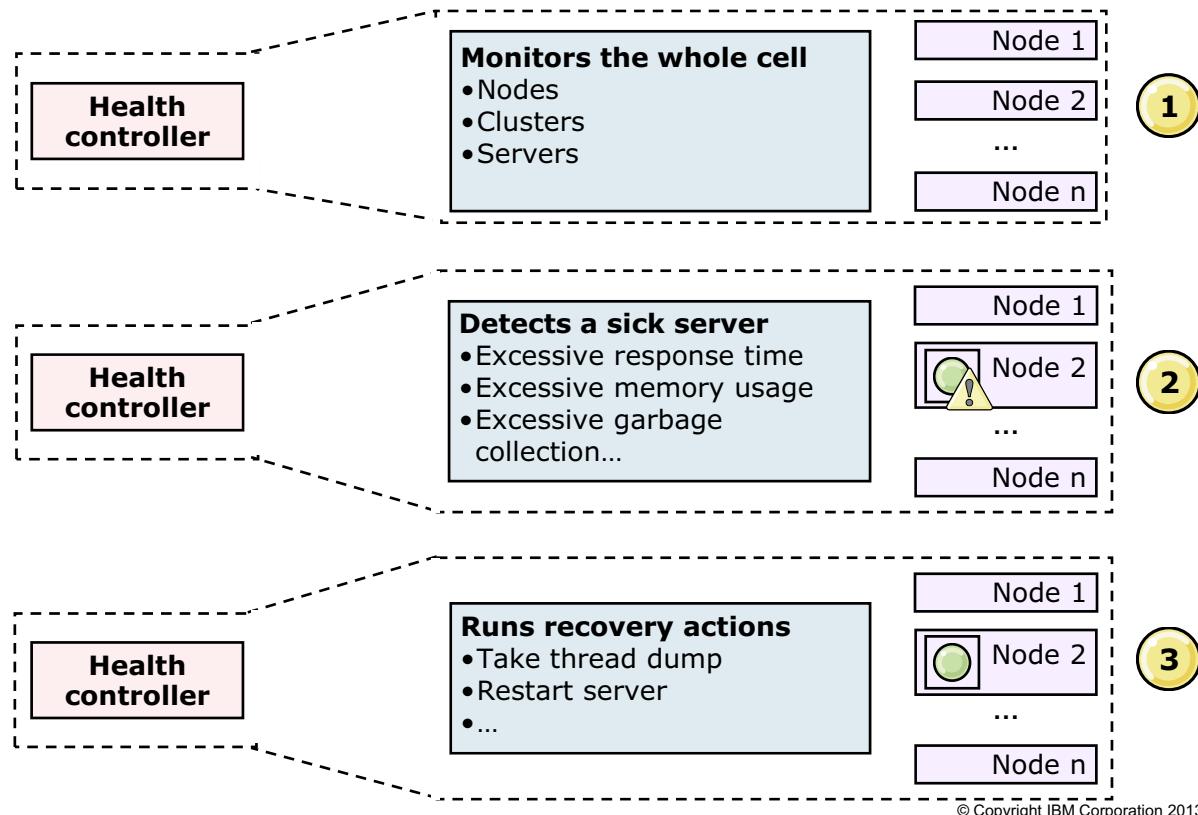
WA5913.0

### Notes:

The Health Management feature of Intelligent Management allows you to monitor your environment and automatically recover when unhealthy servers are found. This feature can be included as part of your complete monitoring solution.

The autonomic manager responsible for monitoring and recovering servers is called the health controller. You define what unhealthy conditions to watch for, and what actions to take with health policies. The health controller works together with the other autonomic managers to ensure that actions are taken intelligently, so service is not interrupted, and service policies are met.

## Health management as part of a monitoring solution



© Copyright IBM Corporation 2013

Figure 17-31. Health management as part of monitoring solution

WA5913.0

### Notes:

Having a good monitoring solution is part of a good problem determination environment. In addition to notifications of when servers are behaving poorly, health management can also automatically recover the sick servers without manual intervention.

## Health policies

- A health policy defines a health condition, reaction, and targets
  - Condition: The **problem state** to look for
  - Reaction: The actions to take when the condition is matched
  - Targets: The resources to monitor
- A health policy also has a reaction mode
  - Automatic
  - Supervised

© Copyright IBM Corporation 2013

Figure 17-32. Health policies

WA5913.0

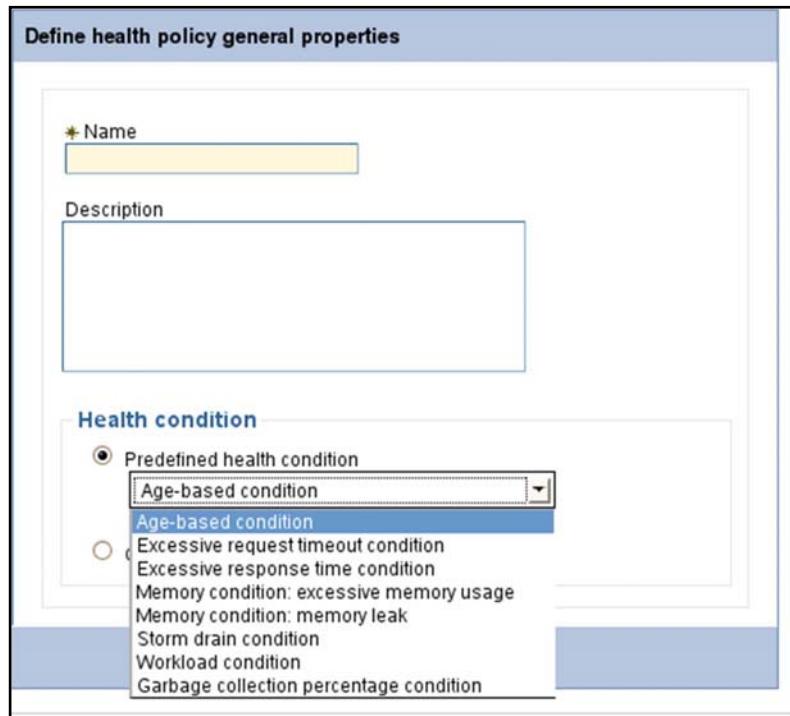
### Notes:

A health policy defines three characteristics. A health condition is what is monitored to determine whether a server is unhealthy. A reaction is the list of actions to take when a health condition is violated. A target is a list of servers to monitor for the health condition.

A health policy can have a reaction mode of automatic – which means the list of actions is taken when the violation is recognized. Or the health policy can have a reaction mode of supervised, which means that administrators must approve the list of actions before they are taken.

## Predefined health policy conditions

- There are a number of predefined health conditions



© Copyright IBM Corporation 2013

Figure 17-33. Predefined health policy conditions

WA5913.0

### Notes:

There are a number of predefined health conditions, which the next couple of slides cover.

## Predefined health policy conditions (1 of 2)

- Age-based
  - Target is running longer than X hours or days
- Excessive request timeout
  - At least X% of requests to the target timeout
- Excessive response time
  - Average response time is greater than X
- Memory condition: excessive memory usage
  - Java heap is X % full for longer than Y minutes
- Memory condition: memory leak
  - Looks for consistent decrease in free memory
  - Three detection levels; slower detection speeds reduce chance of false alarms

© Copyright IBM Corporation 2013

Figure 17-34. Predefined health policy conditions (1 of 2)

WA5913.0

### Notes:

The age-based condition can be used if your server has health problems after being up for some amount of time.

The percentage of requests in the excessive request timeout is calculated based on the number of requests that are sent in 60 seconds. Other settings in your environment control the value that is used to indicate a timeout. See:

<http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=/com.ibm.websphere.wve.doc/ae/welc6topwve.html>

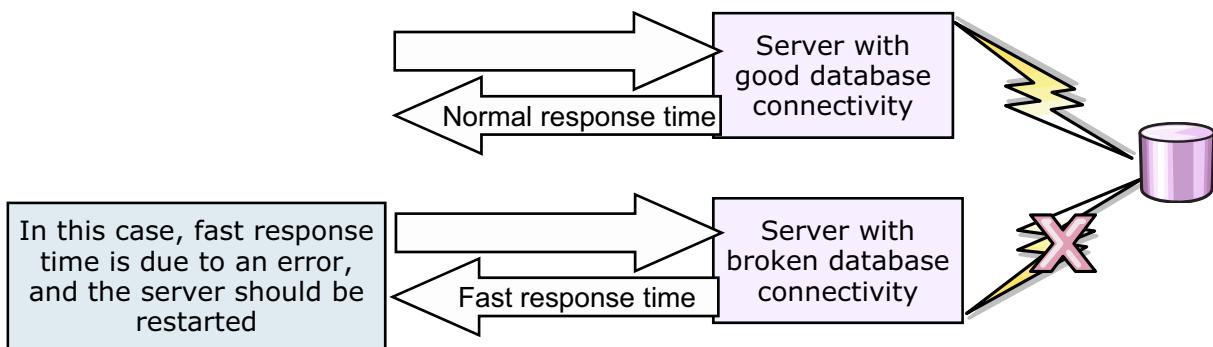
For excessive response time, the average response time that the ODR sees is used. Requests that time out are not included.

The excessive memory usage condition is triggered when the memory usage exceeds a percentage of the heap size for the specified time.

The memory leak condition profiles the JVM heap size after garbage collection occurs and looks for trends of increased consumption. This condition can be set to be more or less aggressive. A more aggressive setting can lead to more false alarms for memory leaks.

## Predefined health policy conditions (2 of 2)

- Workload
  - Target served more than X requests
- Garbage collection percentage condition
  - Whether a JVM spends more than a defined percentage of time in garbage collection during a specified time period
- Storm drain
  - Tracks requests that have a significantly decreased response time
  - Example: An error condition is causing a server to return requests immediately



© Copyright IBM Corporation 2013

Figure 17-35. Predefined health policy conditions (2 of 2)

WA5913.0

### Notes:

The storm drain condition is triggered when requests show a sharp drop in response time from what was measured earlier for those types of requests. Workload management sends more requests to the fastest servers. If a server is returning responses quickly because an error condition is met immediately in the application, this health condition stops all requests from being sent to the problem server.

The workload condition can be used if your applications fail after being sent some particular amount of work.

The garbage collection percentage condition can be used to restart servers if they are spending too much time in garbage collection.

## Custom health policy conditions

- You can define what “unhealthy” means in your environment
- Build custom expressions metrics from:
  - PMI metrics (WebSphere Application Servers only)
  - MBean operations and attributes (WebSphere Application Servers only)
  - ODR metrics (any middleware server)
  - URI and URL return codes (any middleware server)
  - Complex expressions by using a mix of operands are supported
- A subexpression builder is available in the administrative console

© Copyright IBM Corporation 2013

Figure 17-36. Custom health policy conditions

WA5913.0

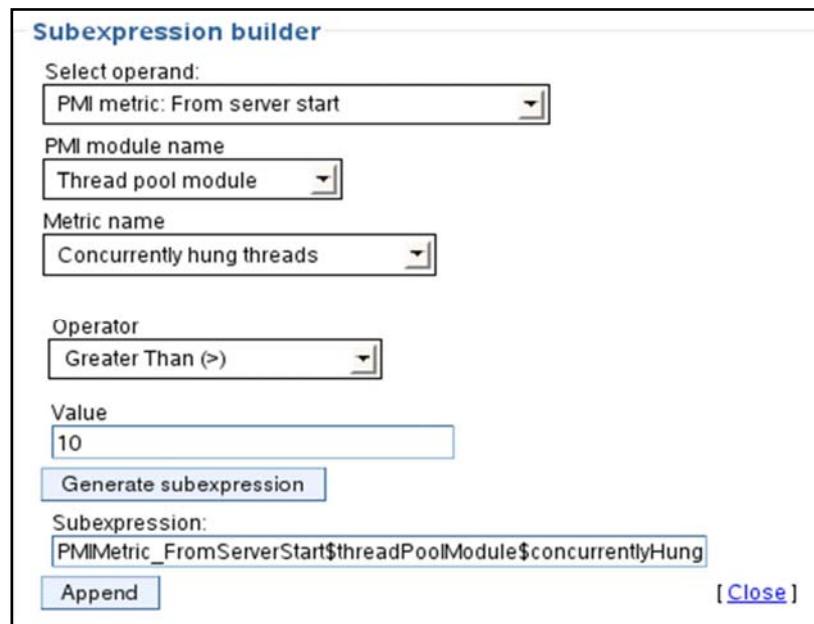
### Notes:

You can also build your own custom health policies. If you can track down why a server failed during a post-mortem of the problem resolution, you might be able to use the information to define a health condition. The new health condition might stop the server from failing for the same reason in the future. You can use metrics and MBeans from the application server to get values to test. The ODR can deliver metrics about how requests are behaving. You can check whether the URLs being sent to a server are getting bad return codes. You can also test whether a return code for any external URL returns a bad value. Combining conditions with logical ANDs and ORs is supported.

Help in building these conditions is available with the subexpression builder in the administrative console.

## Custom health policy conditions

- The subexpression builder helps you to define custom health conditions



© Copyright IBM Corporation 2013

Figure 17-37. Custom health policy conditions

WA5913.0

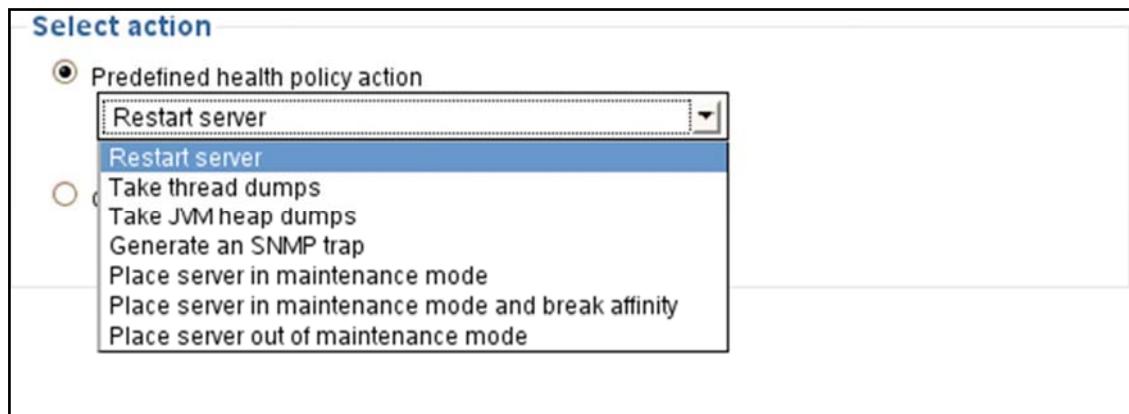
### Notes:

Here is an example of using the subexpression builder to create a condition that is true when the number of hung threads in a server is greater than 10.



## Predefined health policy actions

- There are a number of predefined health policy actions



© Copyright IBM Corporation 2013

Figure 17-38. Predefined health policy actions

WA5913.0

### Notes:

A number of predefined actions can be added to the list of actions to take when a health condition is violated.



## Predefined health policy actions

- Restart server
- Take thread dumps
  - Only WebSphere Application servers that are running with IBM JDK
- Take JVM heap dumps
  - Only WebSphere Application servers that are running with IBM JDK
- Generate an SNMP trap
- Place server in maintenance mode
- Place server in maintenance mode and break affinity
- Place server out of maintenance mode

© Copyright IBM Corporation 2013

Figure 17-39. Predefined health policy actions

WA5913.0

### Notes:

Typically, you restart the server as part of the list of actions. Restarting a server clears up memory, threading, and other problems. You might want to add more actions to the list, such as:

- Put the server in maintenance mode.
- Take a thread dump or memory dump or both.
- Restart the server.
- Put the server out of maintenance mode.



## Custom health policy actions (1 of 2)

Define Custom Action

\* Name

\* Executable

Executable arguments

Name of a variable for referencing a user name in executable arguments

The user name to be substituted for the user name variable at invocation time

Name of a variable for referencing a password in executable arguments

The password to be substituted for the password variable at invocation time

Supported on Operating Systems of type:

windows  
 unix  
 zos

\* Working directory

Defined by selecting **Operational policies > Custom Action**

© Copyright IBM Corporation 2013

Figure 17-40. Custom health policy actions (1 of 2)

WA5913.0

### Notes:

You can also define custom actions to add to the list of actions.

## Custom health policy actions (2 of 2)

- Define the Java class, executable JAR file, or executable program
- Define the arguments
- Optionally define a user name and password to provide
- Define the supported operating system
- Define the working directory

© Copyright IBM Corporation 2013

Figure 17-41. Custom health policy actions (2 of 2)

WA5913.0

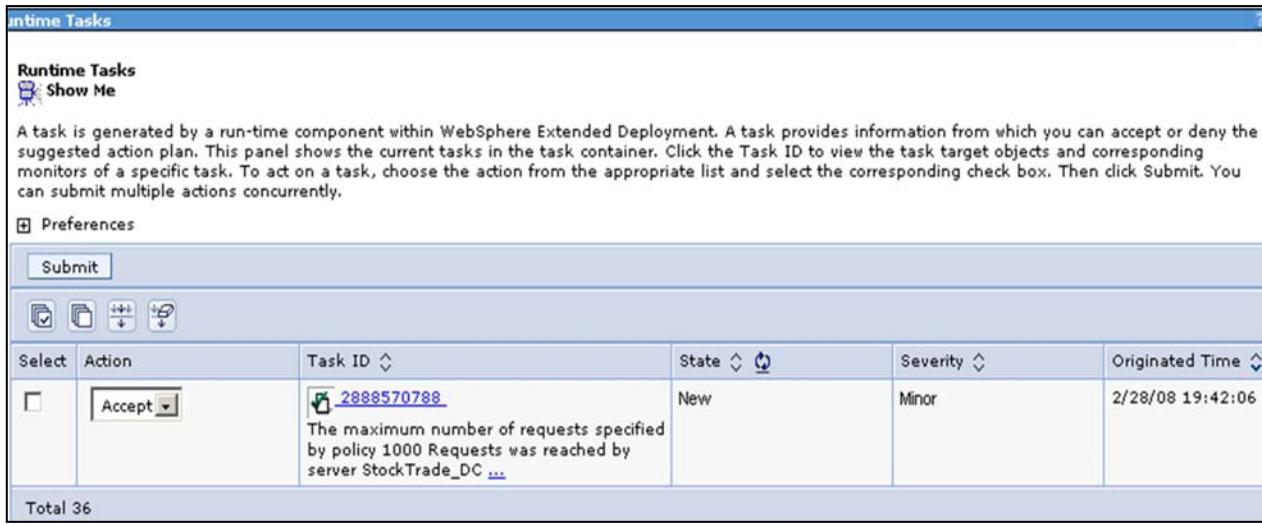
### Notes:

The custom action can be a Java action or a non-Java executable file. You define the arguments to your action, and some other parameters.

 WebSphere Education 

## Health policy reaction mode

- Automatic
  - Health Controller generates a task to carry out the actions
  - The task is automatically accepted and actions are completed
- Supervised
  - Health Controller generates a task to carry out the actions
  - An administrator must accept the task
  - The actions are done



| Select                   | Action | Task ID                                                                                                                            | State | Severity | Originated Time  |
|--------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------|-------|----------|------------------|
| <input type="checkbox"/> | Accept | <a href="#">2888570788</a><br>The maximum number of requests specified by policy 1000 Requests was reached by server StockTrade_DC | New   | Minor    | 2/28/08 19:42:06 |

Total 36

© Copyright IBM Corporation 2013

Figure 17-42. Health policy reaction mode

WA5913.0

### Notes:

A health policy also has a configured reaction mode of automatic or supervise. In both cases, the health controller generates a task that is delivered to the task manager that is running in the admin console. In automatic mode, the actions in the list are performed when the health condition violation is detected. In supervise mode, an administrator must accept the task before the action is taken.

## Cluster restart constraints

- Static clusters: Keep at least one member up
- Dynamic clusters: Placement controller interaction
  - Restart in place
  - Start new instance, stop old instance (placement controller determines location)
  - Stop old instance (if demand is satisfied by remaining servers)

© Copyright IBM Corporation 2013

Figure 17-43. Cluster restart constraints

WA5913.0

### Notes:

Restart actions are done intelligently so that service is not interrupted, and service policies are not violated. For dynamic clusters, the server might be stopped and not started if the resources are not needed, or another server might be started before the sick server is stopped.

## Unit summary

Having completed this unit, you should be able to:

- Describe the basic functions of Intelligent Management
- Troubleshoot Intelligent Management routing problems
- Troubleshoot application placement problems
- Use the health management feature of Intelligent Management
- Use other Intelligent Management features to help with problem determination

© Copyright IBM Corporation 2013

Figure 17-44. Unit summary

WA5913.0

### Notes:

## Checkpoint questions

1. (True or False) Dynamic cluster members are started and stopped as needed to meet demand.
2. Which ODR log contains return codes for errors that the ODR generates?
  - a. proxy.log
  - b. local.log
  - c. cache.log
3. The application placement controller does not start a new cluster member if:
  - a. There is not enough capacity
  - b. It does not improve response time
  - c. All of the above
  - d. None of the above
4. (True or False) Using the centralized logging feature causes all servers in the cell to write their log records to a central location.
5. (True or False) The health management feature can be included as part of a complete monitoring solution.

© Copyright IBM Corporation 2013

Figure 17-45. Checkpoint questions

WA5913.0

### Notes:

Write your answers here:

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

## Checkpoint answers

1. True
2. b. local.log
3. c. All of the above
4. False. Using the centralized logging feature causes all servers in the cell to write their log records to a central location
5. True

© Copyright IBM Corporation 2013

Figure 17-46. Checkpoint answers

WA5913.0

### Notes:

## Exercise 13



Configuring health management policies

© Copyright IBM Corporation 2013

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

8.0

Figure 17-47. Exercise 13

WA5913.0

### Notes:

## Exercise objectives

After completing this exercise, you should be able to:

- Configure a health policy to monitor application servers for the violation of a health condition
- Configure the type of violation to look for
- Configure actions to take when a violation occurs
- Test the health policy by sending requests to the application that trigger the violation of the health policy
- Examine the task that gets generated when a health policy is violated
- Examine the results when the task is completed

© Copyright IBM Corporation 2013

Figure 17-48. Exercise objectives

WA5913.0

### Notes:

# Unit 18. Course summary

## What this unit is about

This unit summarizes the course and provides references for further study.

## What you should be able to do

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study

## Unit objectives

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study

© Copyright IBM Corporation 2013

---

Figure 18-1. Unit objectives

WA5913.0

### Notes:

## Course learning objectives (1 of 2)

After completing this course, you should be able to:

- Use IBM Support Assistant to organize and analyze problem artifacts
- Use problem determination techniques to identify common problems
- Apply problem investigation approaches such as analysis and isolation
- Gather diagnostic data problem artifacts by using administrative tools
- Troubleshoot JVM-related problems such as hung threads, out of memory issues, and crashes
- Use IBM Support Assistant to run tools that analyze diagnostic data
- Identify and troubleshoot common problems with database connections
- Configure and tune database connection pools

© Copyright IBM Corporation 2013

Figure 18-2. Course learning objectives (1 of 2)

WA5913.0

### Notes:

## Course learning objectives (2 of 2)

After completing this course, you should be able to:

- Troubleshoot WebSphere security problems associated with authentication, authorization, SSL, and Java 2 policies
- Identify and resolve Java EE application deployment problems
- Troubleshoot HTTP request flow problems from web server to web container
- Identify and resolve application server startup failures
- Troubleshoot problems associated with WebSphere default messaging and SI bus
- Troubleshoot WebSphere installation problems
- Use Intelligent Management features to configure health policies and tasks
- Communicate effectively with IBM support teams

© Copyright IBM Corporation 2013

Figure 18-3. Course learning objectives (2 of 2)

WA5913.0

### Notes:



## Class evaluation

- Your comments about this class are useful to WebSphere Education
- Feedback on the site, curriculum, and instructor tells WebSphere Education what was good about the class and what can be improved
- Please take the time to complete the course evaluation on the IBM Training website:

<http://www.ibm.com/training/osart>

© Copyright IBM Corporation 2013

Figure 18-4. Class evaluation

WA5913.0

### Notes:



## References (1 of 2)

- IBM Training website:  
[www.ibm.com/training](http://www.ibm.com/training)
- Web pages
  - WebSphere: <http://www.ibm.com/websphere>
  - WebSphere Support page:  
<http://www.ibm.com/software/webservers/appserv/was/support/>
- Redbooks: [www.redbooks.ibm.com/](http://www.redbooks.ibm.com/)
  - SG24-6798-00 *WebSphere Application Server V6 Problem Determination*
  - SG24-7461-00 *WebSphere Application Server V6.1 Problem Determination IBM Redpaper Collection*
  - SG24-7584-00 *IBM WebSphere Application Server V6.1 on the Solaris10 Operating System*
- News groups ([news.software.ibm.com](http://news.software.ibm.com))
  - [ibm.software.websphere.application-server](http://ibm.software.websphere.application-server)
  - See: <https://www.ibm.com/developerworks/community/forums/html/forum?id=11111111-0000-0000-000000000266>

© Copyright IBM Corporation 2013

---

Figure 18-5. References

WA5913.0

## Notes:



## References (2 of 2)

- IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 6 Diagnostics Guide
  - <http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/diagnosis/diag60.pdf>
- The IBM Monitoring and Diagnostic Tools for Java – Health Center

© Copyright IBM Corporation 2013

Figure 18-6. References

WA5913.0

### Notes:

## Unit summary

Having completed this unit, you should be able to:

- Explain how the course met its learning objectives
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study

© Copyright IBM Corporation 2013

---

Figure 18-7. Unit summary

WA5913.0

### Notes:

# List of abbreviations

## A

- AB** ApacheBench  
**AFS** Andrew File System  
**AIX** Advanced IBM UNIX  
**AMI** asynchronous message interface  
**Ant** Another Neat Tool  
**AOT** ahead-of-time  
**APAR** authorized program analysis report  
**APC** application placement controller  
**API** application programming interface  
**ARM** Application Response Measurement  
**ASCII** American Standard Code for Information Interchange  
**AST** Application Server Toolkit

## B

- BSF** Bean Scripting Framework

## C

- CA** certificate authority  
**CCI** Common Client Interface  
**CIM** Centralized Installation Management  
**CIP** custom installation package  
**CMP** container-managed persistence  
**CMS** Certificate Management System  
**CMT** Configuration Migration Tool  
**CORBA** Common Object Request Broker Architecture  
**CP** caching proxy  
**CPU** central processing unit  
**CSlv2** Common Secure Interoperability Protocol Version 2  
**CW** conditioned wait

## D

- DB** database  
**DBA** database administrator  
**DC** domain controller  
**DCS** Distribution and Consistency Services  
**DD** deployment descriptor

**DHCP** Dynamic Host Configuration Protocol

**DLL** Dynamic Link Library

**DMgr** or **dmgr** deployment manager

**DMZ** demilitarized zone

**DN** distinguished name

**DNS** Domain Name System

**DP** diagnostic provider

**DRS** data replication service

**DTD** document type definition

**DTFJ** Diagnostic Tooling Framework for Java

## **E**

**EAR** enterprise archive

**EE** Enterprise Edition

**EIS** enterprise information system

**EJB** Enterprise JavaBean

**EJS** Enterprise Java Services

**EL** Expression Language

**ENC** Enterprise Naming Context

**ESB** Enterprise service bus

**ESI** Edge Side Include

## **F**

**FFDC** first-failure data capture

**FIPS** Federal Information Processing Standard

**FQDN** fully qualified domain name

**FTP** File transfer protocol

## **G**

**GA** generally available

**GB** gigabyte

**GC** garbage collection

**GCD** greatest common divisor

**GCMV** Garbage Collection and Memory Visualizer

**GIF** Graphics Interchange Format

**GMT** Greenwich Mean Time

**GPF** general protection fault

**GTK** GNU GUI Tool Kit

**GSS** Generic Security Services

**GUI** graphical user interface

**H**

**HA** high availability or highly available  
**HACMP** High-Availability Cluster Multi-Processing  
**HAM** high availability manager  
**HFS** Hierarchical File System  
**HP** Hewlett Packard  
**HPEL** High Performance Extensible Logging  
**HP-UX** Hewlett Packard UNIX  
**HTML** Hypertext Markup Language  
**HTTP** Hypertext Transfer Protocol  
**HTTPD** HTTP Daemon  
**HTTPS** HTTP over SSL

**I**

**IBM** International Business Machines Corporation  
**IDDE** Interactive Diagnostic Data Explorer  
**IE** Internet Explorer  
**IIM** IBM Installation Manager  
**IIP** Internet Inter-ORB Protocol  
**IMS** Information Management System  
**I/O** input/output  
**IP** Internet Protocol  
**IPSEC** IP Security  
**ISC** Integrated Solutions Console  
**ISMP** InstallShield MultiPlatform  
**ISV** independent software vendor  
**IT** information technology  
**ITDS** IBM Tivoli Directory Server  
**IVT** Installation Verification Tool

**J**

**J2C** Java 2 Connector  
**J2EE** Java 2 Enterprise Edition  
**J2SE** Java 2 Platform Standard Edition  
**JAAS** Java Authentication and Authorization Service  
**JACC** Java Authorization Contract for Containers  
**JAF** Java Activation Framework  
**JAR** Java archive  
**JASPIC** Java Authentication Service Provider Interface for Containers  
**Java EE** Java Platform, Enterprise Edition

**JAXB** Java Architecture for XML Binding

**JAXP** Java API for XML Processing

**JAXR** Java API for XML Registries

**JAX-RPC** Java API for XML Remote Procedure Calls

**JAX-RS** Java API for XML-based Remote Procedure Calls

**JAX-WS** Java API for XML Web Services

**JCA** Java EE Connector Architecture

**JCE** Java Cryptology Extension

**JCL** Java class library

**JDBC** Java Database Connectivity

**JDE** JD Edwards

**JDK** Java Development Kit

**JHC** Java Health Center

**JIT** just-in-time

**JMS** Java Message Service

**JMX** Java Management Extensions

**JNI** Java Native Interface

**JNDI** Java Naming and Directory Interface

**JPA** Java Persistence API

**JPG** Graphics file type or extension (lossy compressed 24 bit color image storage format developed by the Joint Photographic Experts Group)

**JRE** Java Runtime Environment

**JSF** JavaServer Faces

**JSP** JavaServer Pages

**JSR** Java Specification Request

**JSSE** Java Secure Sockets Extension

**JSTL** JavaScript Tag Library

**JTA** Java Transaction API

**JVM** Java virtual machine

**JVMPPI** Java Virtual Machine Profiler Interface

**JVMTI** Java Virtual Machine Tool Interface

## K

**KB** knowledge base

## L

**LAN** Local area network

**LB** load balancer

**LDAP** Lightweight Directory Access Protocol

**LOA** large object area

**LSD** location service daemon

**LTPA** Lightweight Third Party Authentication

## M

**MAC** message authentication code or media access control

**MAT** Memory Analyzer tool

**MB** megabyte

**MDB** message-driven bean

**ME** messaging engine

**MQ** Message Queue

**MQI** Message Queue Interface

**MVS** Multiple Virtual System

## N

**NAS** network attached storage

**NAT** network address translation

**NFS** network file system

**NIC** network interface card

**NIM** Network Installment Management

**NTP** Network Time Protocol

**NUMA** non-uniform memory access

## O

**OAM** object authority manager

**ODC** on demand configuration

**ODR** on demand router

**OLTP** online transaction processing

**OMG** Object Management Group

**OOM** out-of-memory

**ORB** Object Request Broker

**OS** operating system

**OSGi** Open Service Gateway initiative

**OVF** Open Virtualization Format

## P

**PAR** archive Index File

**PCT** Plug-ins Configuration Tool

**PD** problem determination

**PFBC** properties file based configuration

**PGP** Pretty Good Privacy

**PHD** Portable Heap Dump  
**PHP** personal home page  
**PID** process identifier  
**PKI** public key infrastructure  
**PMAT** Pattern Modeling and Analysis Tool  
**PME** programming model extensions  
**PMI** Performance Monitoring Infrastructure  
**PMR** problem management record  
**PMT** Program Management Tool  
**POJO** plain old Java object  
**PWB** PlantsByWebSphere

## **Q**

**QA** quality assurance  
**QoS** quality of service

## **R**

**RA** registration authority  
**RACF** Resource Access Control Facility  
**RAM** random access memory  
**RAR** resource archive  
**RAS** reliability, availability, and serviceability  
**RC** return code  
**RDBMS** relational database management system  
**RDN** relative distinguished name  
**REST** Representational State Transfer  
**RM** request metrics  
**RMI** Remote Method Invocation  
**RMI/IIOP** Remote Method Invocation Over Internet Inter-ORB Protocol  
**RRA** Relational Resource Adapter  
**RUP** Rational Unified Process  
**RXA** Remote Execution and Access

## **S**

**SAAJ** SOAP with Attachments API for Java  
**SAM** Security Access Manager  
**SAR** SIP application resource  
**SAS** Secure Association Service  
**SBDT** Smart Business Development and Test  
**SCA** Service Component Architecture

**SCADA** supervisory control and data acquisition  
**SCM** source code management  
**SDO** Service Data Objects  
**SDK** software development kit  
**SDLC** systems development lifecycle  
**SFSB** stateful session bean  
**SIB or SIBus** service integration bus  
**SIBDH** Service Integration Bus Destination Handle  
**SIP** Session Initiation Protocol  
**SMTP** Simple Mail Transfer Protocol  
**SOA** service-oriented architecture  
**SOA** small object area  
**SOAP** A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. Usage note: SOAP is not an acronym; it is a word in itself (formerly an acronym for Simple Object Access Protocol)  
**SPI** service provider interface  
**SPNEGO** Simple and Protected GSS-API Negotiation Mechanism  
**SQL** Structured Query Language  
**SSB** stateful session bean  
**SSL** Secure Sockets Layer  
**SSO** single sign-on  
**StAX** Streaming API for XML  
**SUSE** Software und System Entwicklung (German: Software and Systems Development)  
**SVC** supervisor call  
**SVG** Scalable Vector Graphics  
**SWAM** Simple WebSphere Authentication Mechanism

## T

**TAM** Tivoli Access Manager  
**TCP** Transmission Control Protocol  
**TCP/IP** Transmission Control Protocol/Internet Protocol  
**TGC** trace garbage collector  
**TKCARS** Toolkit for Custom and Reusable Solution Information  
**TLS** Transport Layer Security  
**TMDA** Thread and Monitor Dump Analyzer  
**TP** Trade Processor (application in lab exercises)

## U

**UDDI** Universal Description, Discovery, and Integration  
**UNC** Universal Naming Convention

**UNIX** Uniplexed Information and Computing System

**UOW** unit of work

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**UTC** Universal Test Client

**UUID** Universally Unique Identifier

## V

**VM** virtual machine

**VMM** virtual member manager

**VPN** virtual private network

## W

**WAIT** Whole-system Analysis of Idle Time

**WAR** web archive

**WCT** WebSphere Customization Toolbox

**WLM** workload management

**WPM** WebSphere Platform Messaging

**WPS** WebSphere Process Server

**WS** web services

**WS-AT** web services atomic transaction

**WS-BA** web services business activity

**WS-COOR** web services coordination

**WSDL** Web Services Description Language

**WS-I** Web Services Interoperability

**WSIF** Web Services Invocation Framework

## X

**XA** Extended Architecture

**XCT** cross-component trace

**XML** Extensible Markup Language

**XTP** extreme transaction processing

## Y

## Z

**z/OS** zSeries operating system

**ZMMT** z/OS Migration Management Tool

**ZPMT** z/OS Profile Management Tool







**IBM**  
®