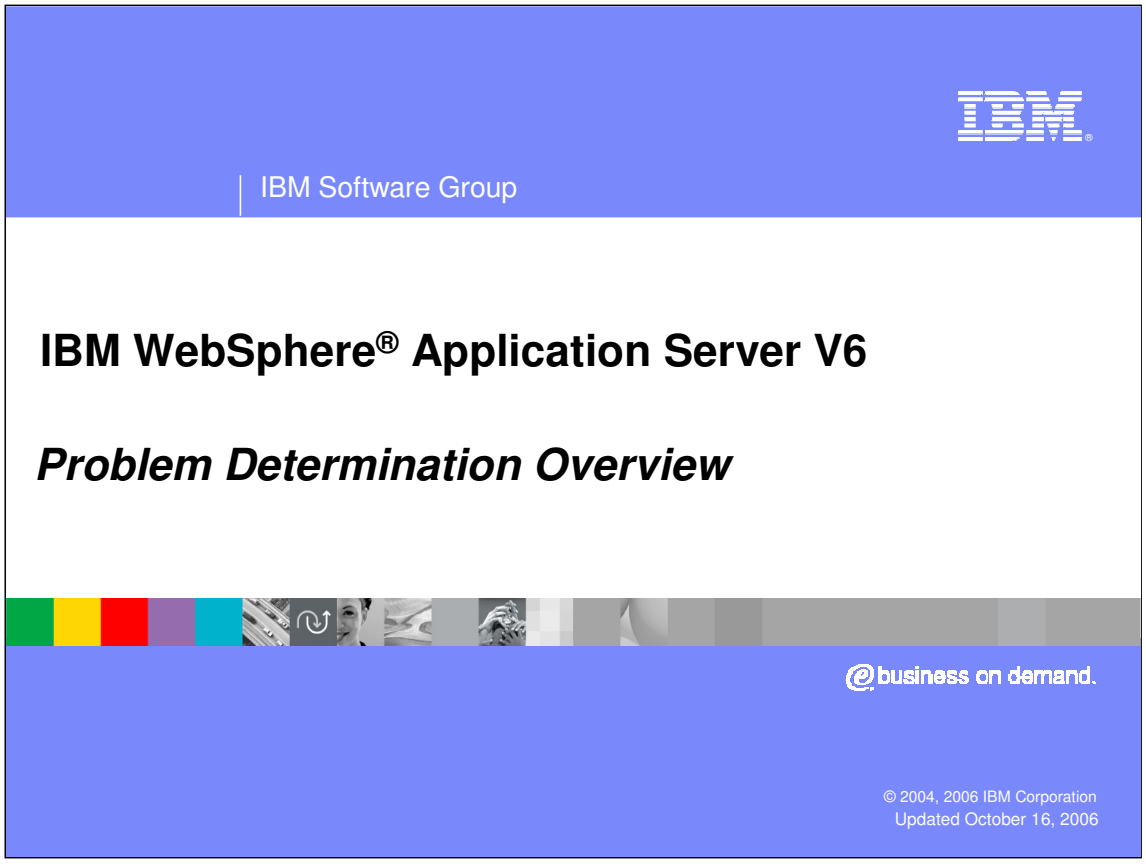


This presentation will focus on problem determination in WebSphere Application Server V6.

Goals

- Understand basic WebSphere Application Server problem determination
  - Where to start
  - How to identify potential causes
- Learn about standard troubleshooting techniques used to isolate problems

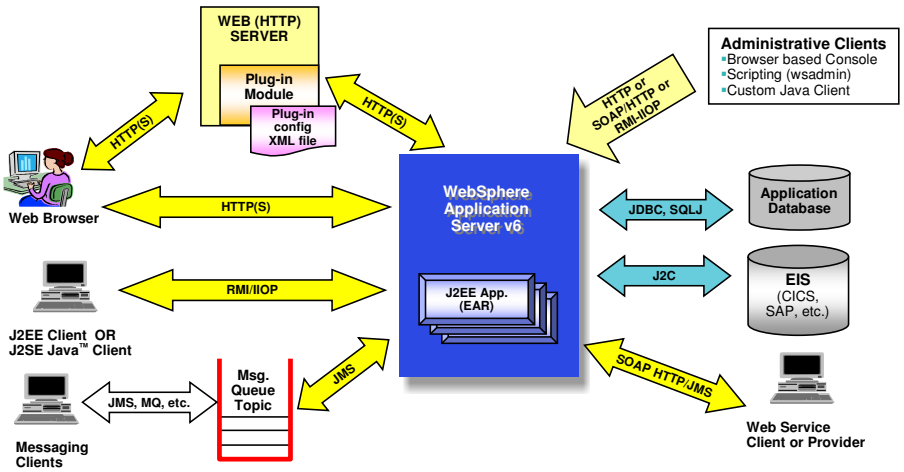
The goals for this presentation are to introduce you to basic problem determination concepts, and also to introduce you to some common techniques for troubleshooting.

Problem Determination Basics

This section covers problem determination basics.

Where Do I Start?

- Consider the big picture initially
  - Understand the architecture / topology



Before you can debug WebSphere Application Server effectively, you will need to understand how it works. You should have some understanding of how the Application Server works, your topology, and how WebSphere Application Server interacts with the other parts of your environment, like database servers. If you are unfamiliar with WebSphere Application Server, you might want to view some of the more basic modules, covering architecture and topology, before continuing with this module.

There are generally several machines or components in a complete environment, and understanding which components are involved with the failure can be difficult for new users. You will likely need to be in communication with the administrators of these related systems to properly debug a complex problem.

As the graphic shows, there are several ways to interact with WebSphere Application Server from a client perspective, and it can also communicate with back-end servers in many different ways.

Initial questions to consider:

- What is the symptom of this problem?
  - for example, a process crashes or users see error messages
- When does the problem occur?
  - for example, nightly, during peak traffic, when a user clicks a particular link, or even intermittently
- Has anything been changed recently?
  - for example, connection pool settings changed or a new version of an application was deployed

Knowing where to look first is often the biggest problem that people have when troubleshooting WebSphere Application Server.

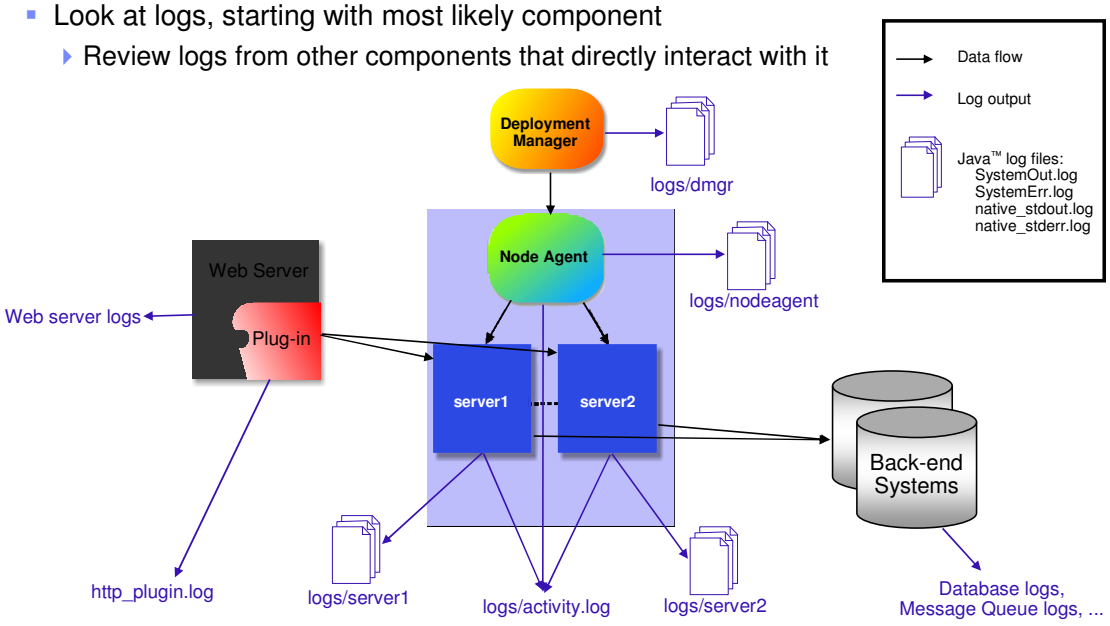
To start, you should ask yourself a few questions that will help you decide where to look: First, what are the symptoms of the problem? If error messages are being displayed to the client, that might contain your first clue. Or if a process has crashed or hung, that is also valuable information. Clearly, if a process has crashed or hung, you will want to start by looking at the logs associated with that process.

Second, when does the problem occur? The timing and frequency of a problem can tell you a lot about the nature of the problem. If, for example, an error occurs every time a user clicks a link to view an “account status” page, you can start your investigation by looking into the actions performed by the “account status” code. Similarly, if a problem only occurs when user traffic is high, it is likely that you have constrained resources at some point in your system. Intermittent problems are generally the hardest to solve, because they can appear to be unpredictable.

Third, have any changes been made to the system recently? Have there been any configuration changes or application updates? If something has changed, it is a good idea to start by investigating that component, particularly if the problem started happening in the same time frame.

Log Files

- Look at logs, starting with most likely component
  - Review logs from other components that directly interact with it



The first step in problem determination is usually to take a look at the WebSphere Application Server logs. Based on the questions from the previous slide, think about the component that is most likely involved with the failure. Each component writes data to at least one log file, and you may have to look at several log files to get a complete picture of the failure. While there can be several pieces to a complete production environment, this lecture will be specifically discussing logs created by WebSphere Application Server itself. Even though they will not be covered here, logs from other systems can be important (for example, Web server, database, or firewall logs). Consult your product's documentation for information on its log files. There are also many log files that are not shown in this picture. The logs shown in this picture are the logs that you should look at first when dealing with a problem during runtime. The other logs will be discussed later. It is also a good idea to synchronize the clocks on your systems, so that it will be easier to correlate error messages on one machine to related messages on another machine.

Installation Logs: Overview

Group	Log Files (default names)	Description
Installation logs	log.txt	Main installation log
	lhs_log.txt	Main IBM HTTP Server (IHS) installation log
	WASPreUpgrade.log, WASPostUpgrade.log	Log the actions of the migration tools that backup and restore existing configuration

Found in <INSTALL_HOME>/logs

During installation, logs are created in the “logs” directory underneath the root installation directory. “log.txt” is the main installation log, and should be the first place you look if installation fails. If the installation program cannot get far enough to create log.txt in this location, log.txt will be created in your system’s temporary directory.

Runtime Logs: Overview

Group	Log Files (default names)	Description
Service log	activity.log	Binary log file contains data from each JVM in a node, for analysis using Log Analyzer
JVM logs	StandardOut.log, StandardErr.log	Contain all messages sent to the Java System.out and System.err buffers.
Native process logs	native_stdout.log, native_stderr.log	Contain messages sent to stdout and stderr from native code segments, including the JVM
Embedded HTTP server logs	http_access.log, http_error.log	Contain all requests to the embedded HTTP server
HTTP server plug-in log	http_plugin.log	Contains data about the operation of the HTTP server plug-in module
Command-line program logs	startServer.log, addNode.log <*command*>.log	Contain data about the execution of individual command-line utilities.
System application & sample application logs	<*name*>_deploy.txt, <*name*>_config.txt	Deployment and configuration logs for each of the enterprise applications installed by the WebSphere Application Server installer (Administrative Console, samples, etc),

Found in <PROFILE_HOME>/logs by default

The logs shown on this page will exist for each profile, and as such will exist in the particular profile's logs directory. Other than activity.log, which only exists once per node, each server on your node will have its own copy of these runtime log files.

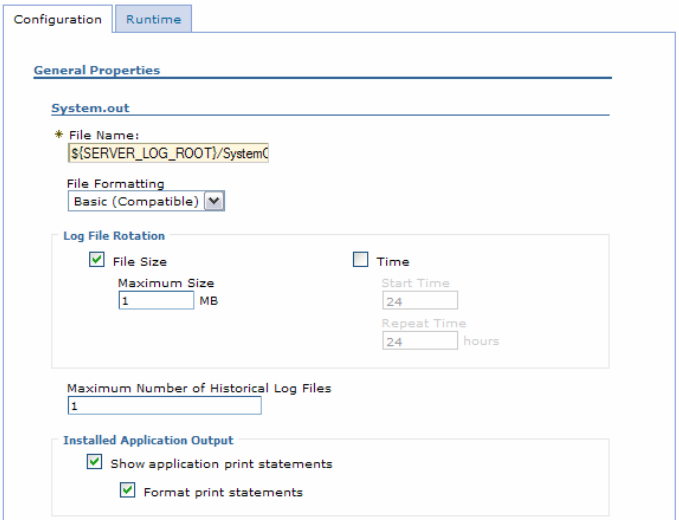
The JVM logs are the most useful of the runtime logs. They contain information about the server runtime and any messages that the application writes to System.out or System.err. You will also find exception and stack trace information in these logs. The native process logs contain information that gets logged by native code, such as the JVM itself, or some parts of the security implementation.

The other logs shown in this table are relatively self-explanatory; they correspond to a particular component or command-line program.

Configuring Java™ Virtual Machine (JVM) Logs

From Servers > Application Servers > *servername*:

- Logging and Tracing > JVM Logs
- System.out and System.err logs configured from here
- Logs are self-managing
  - Can rollover based on time or file size
  - Number of historical log files is configurable



The JVM logs, System.out and System.err, can be configured by clicking “Logging and Tracing”, then “JVM Logs” from your Application Server’s main configuration page. The options available on this page are the same options that were available in V5, including size-based and time-based rollover of log files.

Basics: Beyond Log Files

- Component isolation
  - Take components out of equation (logically) to determine which is failing
- Tracing
  - Logs detailed runtime information about a component
  - Useful when a component doesn't seem to be working properly
- Java thread dumps
  - Provide detailed information about all active Java threads
  - Useful in several situations, most notably for hung servers

Log files are a great starting point, but sometimes they do not exactly pinpoint the cause of the problem. You should use the log files to gather data about the problem, then drill down using some of the techniques described here. Each of these topics will be discussed further toward the end of this presentation.

Component isolation is the practice of trying to simplify your environment by removing components one at a time to see if the problem still occurs.

WebSphere Application Server has comprehensive diagnostic tracing facilities. Trace information can be useful when you know the general area where a problem occurs. You can then run a trace to view exactly what the component is doing while the error occurs.

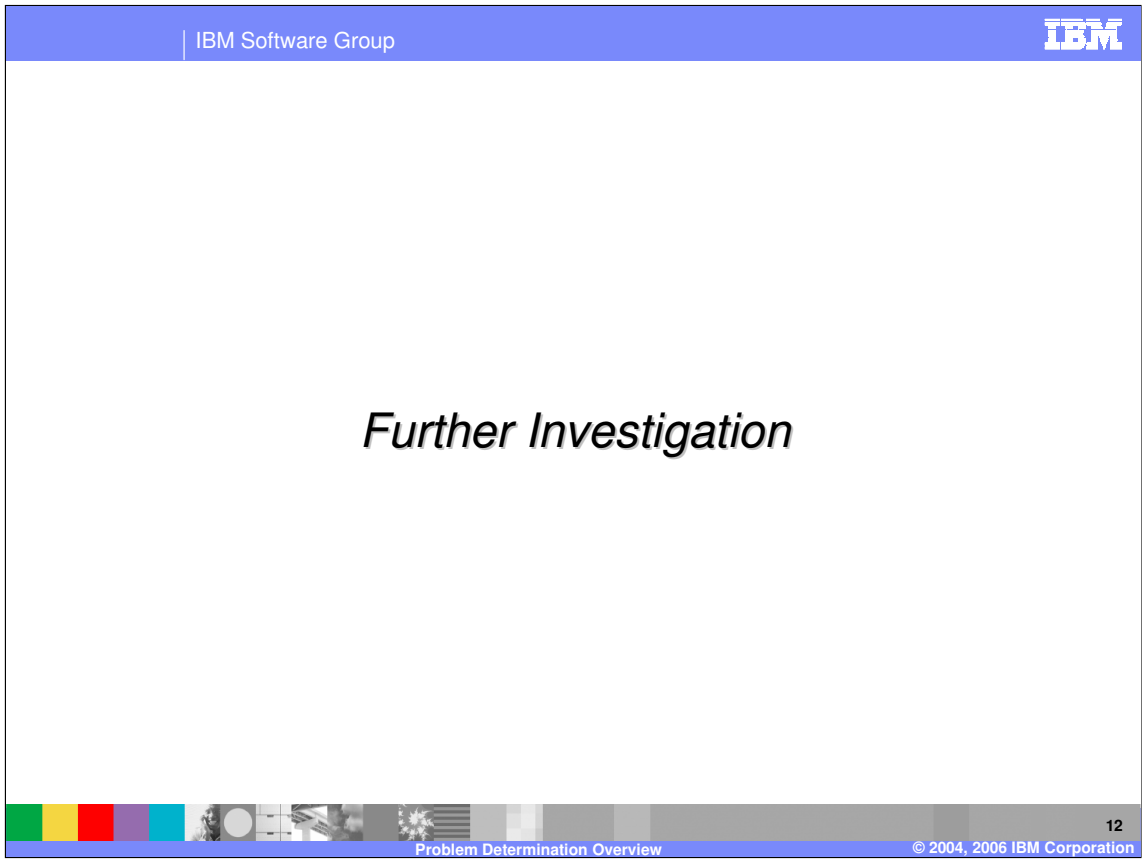
Java thread dumps, also called Java dumps, are snapshots of the state of the Java Virtual Machine at a given point in time. These can be helpful in several situations because the dump will show you what each individual thread is doing, and the state of locks and monitors. Java dumps are critical for debugging hung servers.

Basics: Beyond Log Files (cont.)

- Java heap dumps
  - Provide detailed information about objects in the Java memory space
  - Useful for debugging memory leaks
- Profiling/debugging
  - Observe exactly what the application is doing
  - Useful for debugging application problems

Java heap dumps give you a complete look at every object that is populating the Java memory space, called the “heap”. You can see the size and location of every object, and also view relationships between objects. Heap dumps are most useful for diagnosing Java memory leaks.

Potentially the most complicated techniques are profiling and debugging. Profiling is the practice of enabling software probes within the JVM to observe the behavior and usage patterns of your application. Debugging allows you to step through your code in an orderly fashion, and observe the state of the application at any given time. Both of these techniques require complementary products, such as IBM® Rational® Application Developer.



This section discusses ways to further troubleshoot your WebSphere Application Server problems, beginning with component isolation.

Component Isolation

- Can be helpful to simplify your environment
- The more factors you can eliminate, the better
- When there is a communication problem between two components, test them separately

One way to drill down and isolate the source of the problem is to logically remove components from your environment. Simplifying the scenario will make the problem easier to identify. One situation where such simplification can be useful is when two components are not communicating with each other. Testing each component separately can shed some light on the issue.

Component Isolation: Examples

- Application Server not communicating with database
  - Try connecting to the database with another Java Database Connectivity (JDBC) client
    - for example, JDBCTest, available on IBM support Web site, tests JDBC connectivity
- Web clients not getting response from Application Server
  - Try logically removing the Web server and plug-in from the environment
    - Query the Application Server directly using the embedded HTTP server, bypassing the Web server and plug-in
- Consider firewalls

For example, if an Application Server is having trouble communicating with a database, try querying the database with a different JDBC client. A tool called JDBCTest is available on the WebSphere Application Server support website for just this purpose. If you cannot query the database with JDBCTest, then you have eliminated the Application Server as the source of the problem. The network, database or the JDBC driver are likely culprits.

Similarly, if web clients are not getting responses from an Application Server, try logically removing the web server from the scenario. Query the application server directly using the embedded HTTP server, which is listening on the port configured for the web container channel's thread pool. If the Application Server is still not responding, then your Application Server is not functioning properly. If, however, you do get a response, then you should investigate the web server and the plug-in.

In situations like this, remember to think about any firewalls that might be in between the systems in question. If possible, it can be helpful to temporarily remove the firewall, to eliminate it as a potential culprit.

Diagnostic Tracing

This section discusses diagnostic tracing.

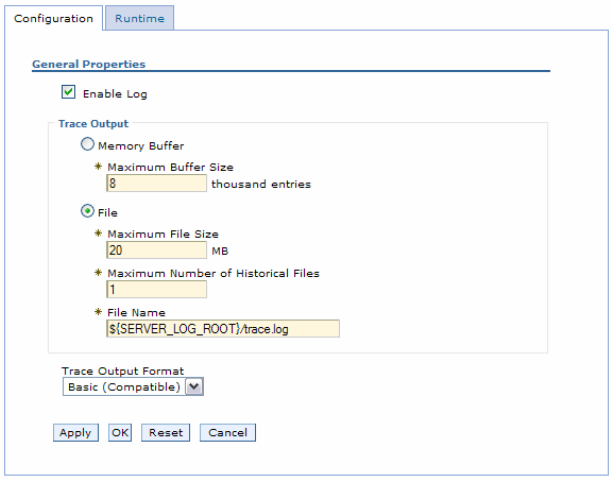
Tracing

- Enables detailed logging of runtime actions
- Granular
  - You can trace specific components
- Useful when you have an idea of where the problem might lie, but you need more information
- Performance impact can be very high
- Can quickly take up large amounts of disk space

Diagnostic tracing logs detailed messages about the actions of the runtime. It lets you observe WebSphere Application Server's internal behavior. Since you can specify particular components to be traced, tracing can be useful for getting more information once you know that a problem exists in a particular area. Because tracing logs so much data, it can quickly take up a lot of disk space, and has a negative impact on performance. This means that tracing is usually not recommended in a production environment.

Enabling Trace

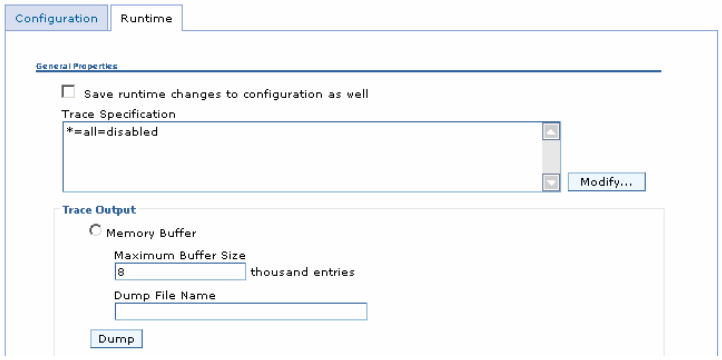
- Logging and Tracing > Diagnostic Trace
- “Configuration” tab modifies configuration settings and will take affect at the next startup
- “Runtime” tab affects the currently running server
- More information in the “Logging and Tracing” presentation



Tracing can be activated by clicking on **Logging And Tracing**, then **Diagnostic Trace** on an Application Server's main page. As in previous releases, you can activate a trace on a running server or you can choose to trace the startup of a server. Use the Configuration tab to configure the trace settings for the next startup of the server. Use the Runtime tab to activate tracing or dump the trace buffer on a running server. More information on configuring tracing can be found in the presentation titled “Logging and Tracing”.

Tracing: Using the memory buffer

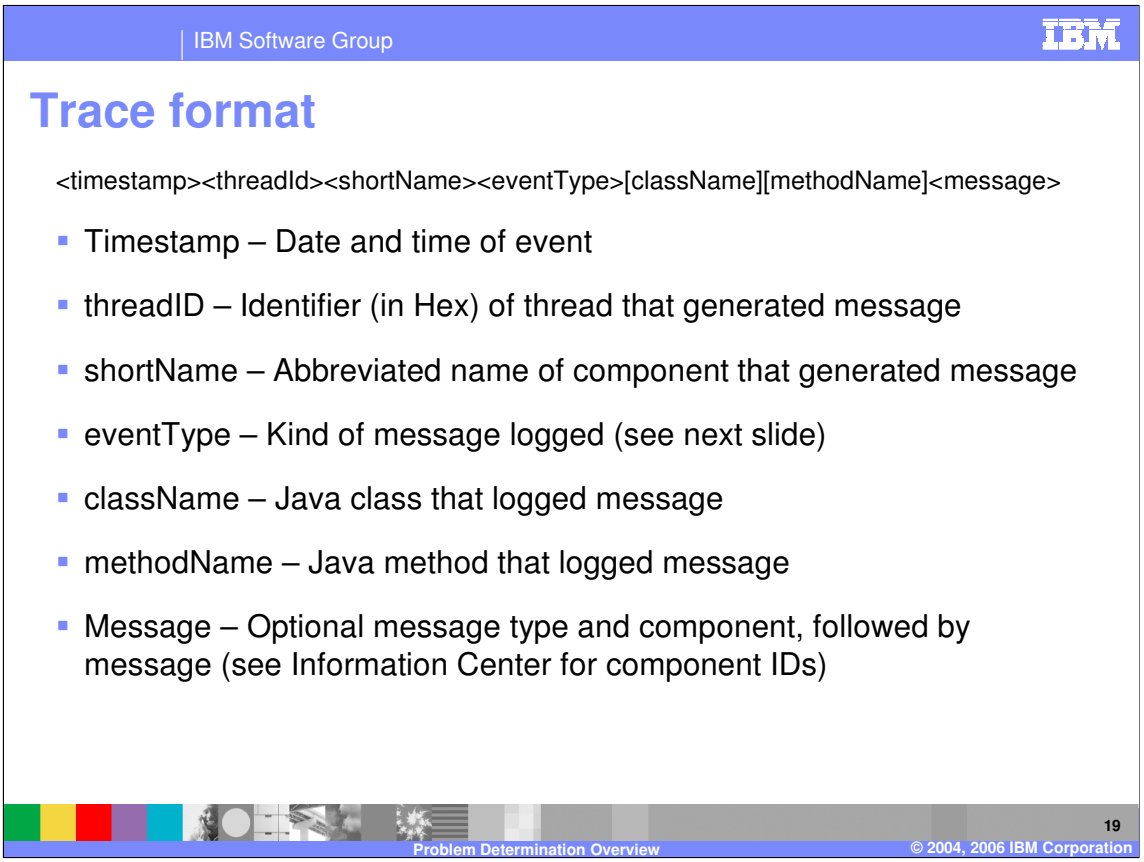
- Can reduce performance overhead
- Press the “dump” button to dump the buffer contents to a text file



If the performance penalty of tracing is too great, you can use the memory buffer to capture a set amount of trace data. Performance is not affected as heavily because trace data is going into the memory buffer, rather than constantly being written to disk. You can specify the number of trace entries that the buffer should contain. When the buffer fills, the oldest trace entries will be purged.

You can press the “Dump” button on the Runtime tab at any time to dump the current contents of the buffer to a text file.

This is most useful when you want to capture trace data for a short-running test. It is not practical for longer tests, because it will not capture data from the entire test.



Diagnostic tracing uses the format shown here. The trace format is similar to the JVM log format.

Some of the most interesting messages to look for in a trace are the entry and exit messages. These messages track entry and exit into Java methods within the traced component. This allows you to watch the code-execution path through the component. In addition to being helpful in problem determination, these messages can be a great educational resource.

Trace Format: Event Types

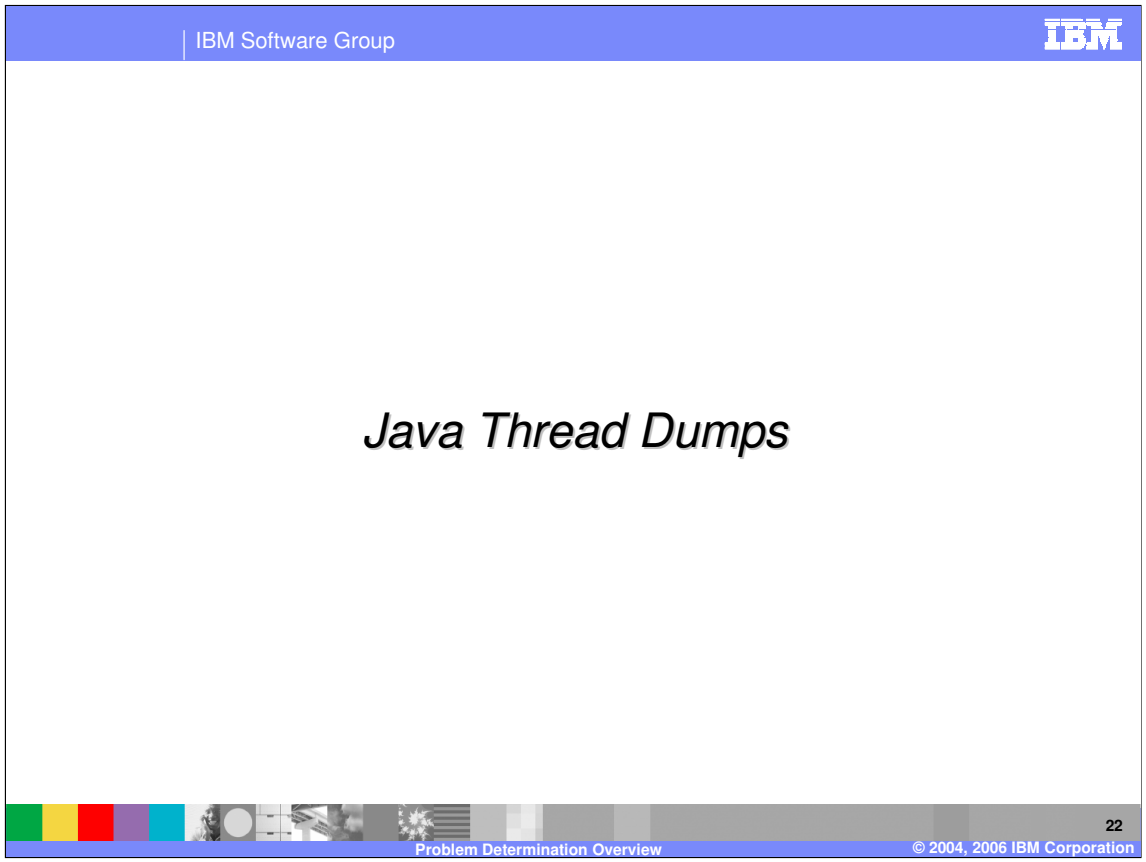
ID	Description	ID	Description
F	Fatal Message	E	Error Message
W	Warning Message	A	Audit message
I	Informational message	C	Configuration change message
D	Detail message	1	Fine detail
2	Finer detail	>	Method entry
<	Method exit	3	Finest detail

The event types shown here are used by diagnostic tracing.

Trace Format: Event Types

ID	Description	ID	Description
1	Fine detail	2	Finer detail
3	Finest detail	A	Audit message
D	Detail message	I	Informational message
E	Severe message	C	Configuration change message
F	Fatal message	W	Warning message
>	Method entry	<	Method exit

The event types shown here are used by diagnostic tracing.



This section covers Java thread dumps.

Java Thread Dumps

- Contain information about the state of the JVM at a given time
  - Thread state information for all threads
  - Lock/monitor information
  - Other environment data

Thread dumps are snapshots of the state of the JVM at a given time. They show what every thread was doing at the time of the dump, the state of monitors, and other information about the JVM.

Java Thread Dumps: Benefits

- Debugging hung processes
  - Compare multiple dumps to observe threads over time
  - Threads that do not change may be hung
  - Use the lock/monitor information to determine if two threads are deadlocked.
- Finding performance bottlenecks
  - If there are always several threads in a particular method, that method might be causing a bottleneck
  - Investigate code path to determine if method can be optimized

Thread dumps are most useful for debugging hung processes. If threads do not appear to change over the course of several dumps, those threads may be hung. In most situations, comparing three dumps taken three minutes apart gives a good window of data. You can also determine if two threads are deadlocked by looking at the monitor information. A “deadlock” is when two threads are each waiting for a monitor to be released by the other. Neither thread can proceed, since they are waiting on each other, and they will hang indefinitely.

Thread dumps are also useful for finding performance bottlenecks. If there are frequently several threads in a given method, that method is likely a bottleneck. If there are several threads waiting for the same resource, that resource is constrained, and you should investigate if access to that resource can be parallelized.

Java Thread Dumps

- Generate a dump by sending SIGQUIT (kill -3) to the JVM process (on UNIX®)
- On all platforms, use wsadmin to generate a dump:

```
> set jvm [$AdminControl completeObjectName type=JVM,process=servername,*]
> $AdminControl invoke $jvm dumpThreads
```

- Text file created in <PROFILE_HOME>/bin directory
  - File named javacore<identifier>.txt, where <identifier> contains timestamp and PID
  - Ensure you have proper permissions and disk space in the specified directory

To specify an alternate location for the dump file, set the IBM_JAVACOREDIR environment variable.

The wsadmin command shown here is particularly useful on Windows®, because you cannot generate a dump by sending a signal to an arbitrary process, as you can under Unix®.

For more information on how to read thread dumps, refer to the Java Diagnostics Guide, listed at the end of this presentation.

Thread dumps from a WebSphere Application Server process cannot be generated in this manner on z/OS®.

Thread Analyzer

- Gathers and analyzes thread dumps from an Application Server
- Use to analyze the following
  - Performance bottlenecks due to either configuration or application problems
  - Determining if threads are being blocked on monitors
  - Determining percentage of work being done on the Application Server
- Provides recommendations based on analysis

Thread Analyzer is a graphical interface for reading and analyzing Java thread dumps. It allows you to easily view the number of threads that are in a particular method, or compare the number of threads currently doing work to the number of idle threads.

It also performs analysis on the thread dump to help determine where problems might exist. For example, it can locate deadlocks and highlight potential bottlenecks.

Thread Analyzer is also capable of requesting a thread dump from a running server. Sometimes, if a server is hung and not responding to kill -3 to generate a dump, Thread Analyzer might be able to produce a dump, because it uses a different method to request it.

Thread Analyzer is supported on Windows, Unix, and Linux®. It can be downloaded from IBM using the link at the end of this presentation. The tool is not supported on z/OS, since WebSphere Application Server for z/OS does not create traditional thread dumps.

Java Heap Dumps

This section covers Java Heap Dumps.

Java Heap Dumps

- Print a record of all objects in the Java heap to a text file
- Raw heap dump file contains size, address, and references for each object
- Garbage collection is forced before a dump
  - This is a recent enhancement, added in JVM 1.4.1 SR1
- Compare heap dumps over time to determine if memory is leaking

Heap dump capability is a feature of the IBM Virtual Machine that prints a record of all objects in the Java heap to a text file. The size and address of each object are recorded and the addresses of all objects that it references. This information can help you understand which objects are responsible for taking up large amounts of memory.

Recently, the JVM was enhanced to trigger a garbage collection before dumping the heap information. This makes the dump more accurate, because only “live” objects (that is, objects that still have references to them) will appear in the dump.

By taking multiple heap dumps and comparing object sizes over time you can determine if a particular object is not being dereferenced properly, which could result in a memory leak.

Note that heap dumps are a feature of the IBM Virtual Machine. To diagnose memory problems on platforms that do not use the IBM VM, you will need to use a JVMPI profiler, such as Hprof.

Heap Dump Analysis Tools

- Tools make interpreting heap dumps easier
- Reading of a raw heap dump is not practical
- HeapRoots is a free tool for reading heap dumps
  - http://www.alphaworks.ibm.com/tech/heaproots

Heap dump files can be very large and hard to interpret in their original format. There are several tools that can help you find the useful information in a heap dump quickly. One such tool, HeapRoots, is freely available from IBM AlphaWorks.

Java Heap Dumps: Usage

- Automatically generated whenever a java.lang.OutOfMemoryException is thrown
- To enable manual heap dump generation, set the environment variable “IBM_HEAPDUMP=true”
  - “IBM_JAVA_HEAPDUMP_TEXT=true” enables text-format heap dumps
- Send a SIGQUIT (kill -3) to the JVM to generate a heap dump (UNIX)
  - Or use the same wsadmin command that generates thread dumps (All platforms)

By default, the IBM VM will produce a heap dump whenever a java.lang.OutOfMemoryException is thrown. Heap dumps can also be generated at any time if the environment variable IBM_HEAPDUMP=TRUE is set. With this variable set, sending a SIGQUIT to the JVM, or triggering a Java dump with wsadmin will cause a heap dump to be generated as well.

Java Heap Dumps: Usage (cont.)

- heapdump<identifier>.txt will be created in <PROFILE_HOME> directory
  - <identifier> represents process ID and timestamp
    - <identifier> format varies from platform to platform
  - File location can be overridden by IBM_HEAPDUMPDIR environment variable
  - Ensure you have proper permissions and disk space in this directory

By default, the heap dump file will be created in your profile's home directory. The identifier shown in the filename above represents the process and the timestamp, but the exact format varies from platform to platform.

The file location can be overridden by the IBM_HEAPDUMPDIR environment variable, but you should ensure that you have permissions to write to the specified directory and that there is enough free space to hold the heap dump file.

Profiling/Debugging

- Useful for troubleshooting application problems
- Profiling tools enable you to observe your application's behavior and locate performance problems
  - Resource utilization: processor, memory, etc
  - Time spent in a particular method
- Debugging tools help you step through application code to locate bugs
- IBM Rational® Application Developer V6 includes profiling and debugging tools

Profiling and debugging are two techniques for troubleshooting application problems.

Profiling tools enable you to look inside of a running application to characterize its behavior, generally from a performance perspective. You can observe the application's use of resources, like processor time or memory, and you can also observe the amount of time spent in each method, to determine if a particular method is causing a slowdown. Java Profilers use JVMPI, the Java Virtual Machine Profiling Interface, to gather this data. A simple profiling tool, called Hprof, is built into the Java Virtual Machine.

Debugging tools are generally used to find bugs in an application. They allow you to step through the execution of code in a controlled fashion, making it easier to isolate and eliminate problems.

IBM provides profiling and debugging functionality in Rational Application Developer. Several third-party tools are also available, and fill a wide variety of needs in this space.

Summary

- Always start by thinking of the big picture
  - Problems can be caused by any component
- Log files are your first clues
- Based on what you learn from the logs, there are several ways to dig deeper:
  - Tracing
  - Java thread dumps
  - Java heap dumps
  - Profiling/debugging

In summary, this presentation has given you a basic overview of problem determination tactics for WebSphere Application Server. Always remember to look at the big picture, and consider that any component could be the cause of your problem. Start troubleshooting by looking at the relevant log files. The log files should get you started, and you can dig deeper by employing the tactics discussed in this presentation. Component isolation helps you simplify the environment and drill down to the broken component. Tracing logs detailed information about the execution of WebSphere Application Server. Java thread dumps show you the state of every thread in the Java runtime at a given time. Java heap dumps are a snapshot of the Java memory space at a point in time. Profiling and debugging are methods by which you can use external tools to learn more about the execution of your code and the server.

Additional Resources

- WebSphere V6 Information Center:
  - http://www.ibm.com/software/webservers/appserv/infocenter.html
- WebSphere support website: Search Technotes, tools, interim fixes, and fix packs
  - http://www.ibm.com/websphere/support
- IBM Java SDK 1.4.1 Diagnostics Guide: excellent troubleshooting information
  - http://www.ibm.com/developerworks/java/jdk/diagnosis/diag141sr1.pdf

This slide lists some useful problem determination resources.

Tools Reference

Tool	Purpose	Download Link
Thread Analyzer	Analysis of Java thread dumps	http://www.ibm.com/developerworks/websphere/downloads/thread_analyzer.html
HeapRoots	Console-based analysis of Java heap dumps	http://www.alphaworks.ibm.com/tech/heaproots
HeapWizard	Graphical analysis of Java heap dumps	ftp://ftp.software.ibm.com/software/websphere/info/tools/heapwizard/
ClassLoaderViewer	Admin console plug-in for displaying the classloader hierarchy	http://www.ibm.com/developerworks/websphere/library/techarticles/0312_cocasse/0312_cocasse.html
JDBCTest	Tests connectivity to DB2 and Oracle databases using JDBC	ftp://ftp.software.ibm.com/software/websphere/info/tools/jdbctest

This table lists several tools that you might find useful. Some of them have been discussed in this presentation. All of them are available free of charge from IBM, at the specified locations.

Trademarks, Copyrights, and Disclaimers

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

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

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

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

Linux is a registered trademark of Linus Torvalds.

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

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

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