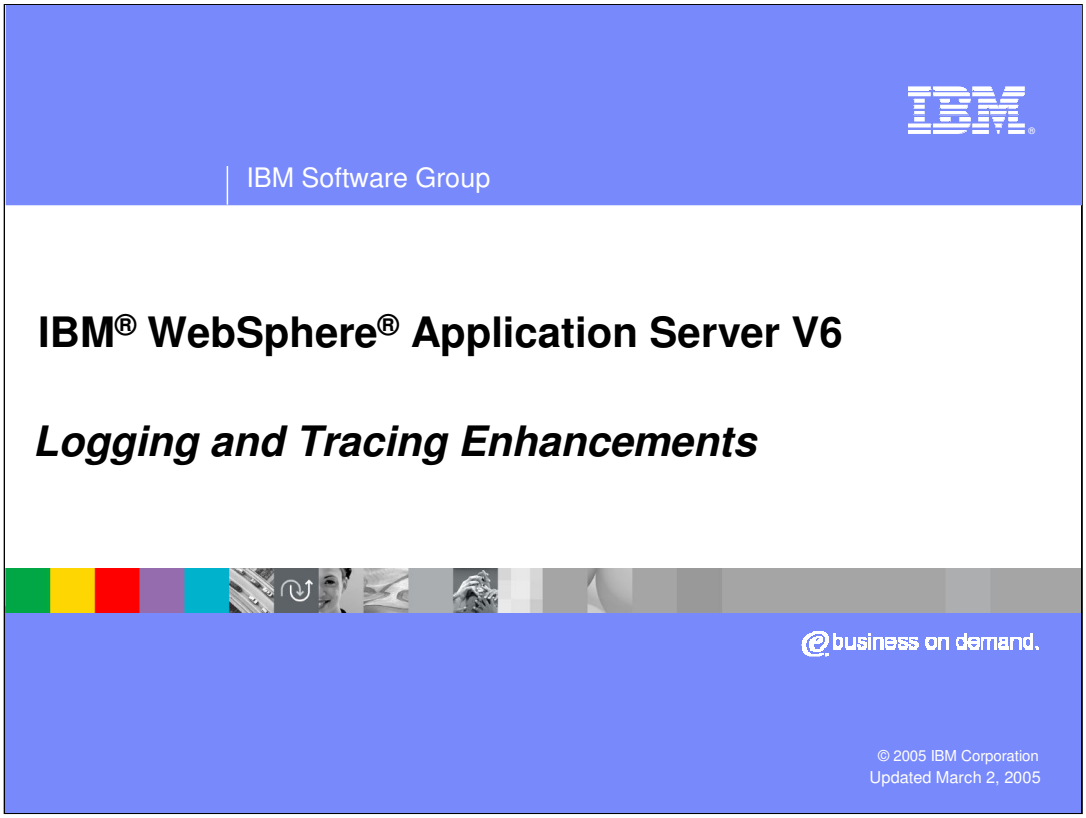


This presentation will focus on the changes to logging and tracing in WebSphere Application Server V6.

Goals

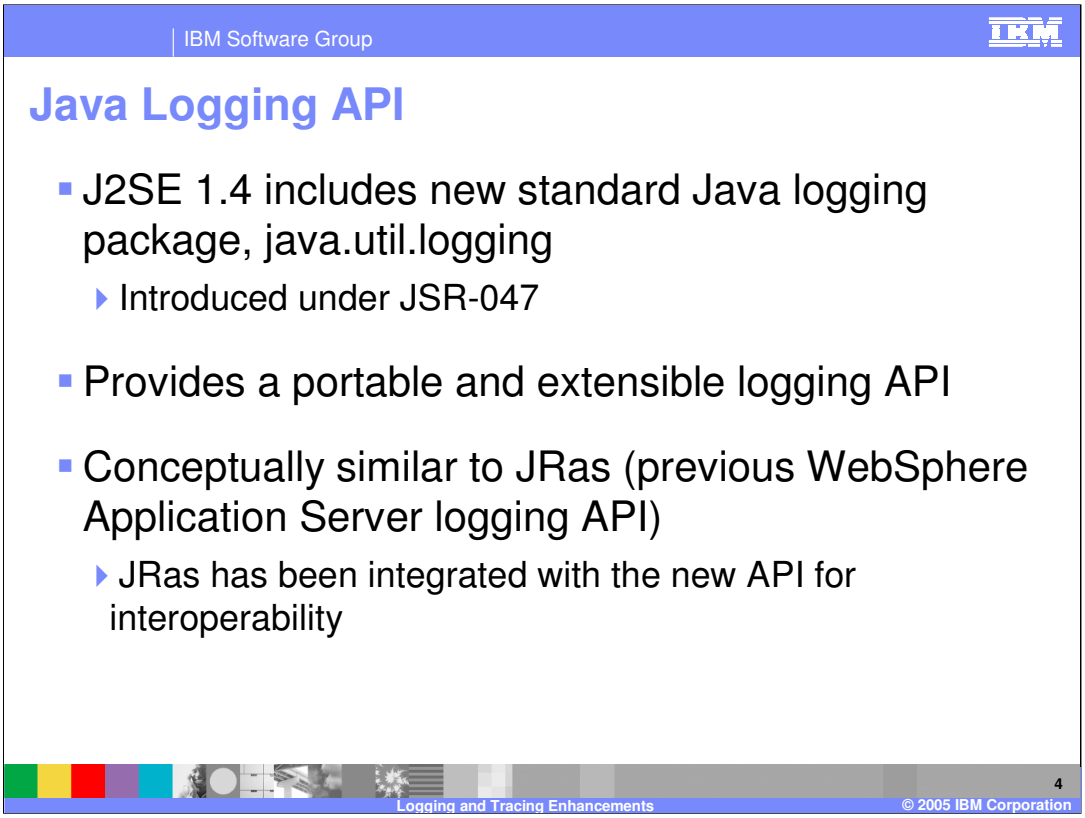
- Introduce the new logging and tracing architecture in WebSphere Application Server V6.0
  - Java™ logging Application Programming Interface (API)
  - Configuring logging and tracing
  - Embedded HTTP server access and error logging

The goal of this presentation is to introduce the new logging and tracing architecture in WebSphere Application Server V6. You will become familiar with the Java logging API, and then you will learn how to configure logging and tracing in V6.

Section

Java Logging API

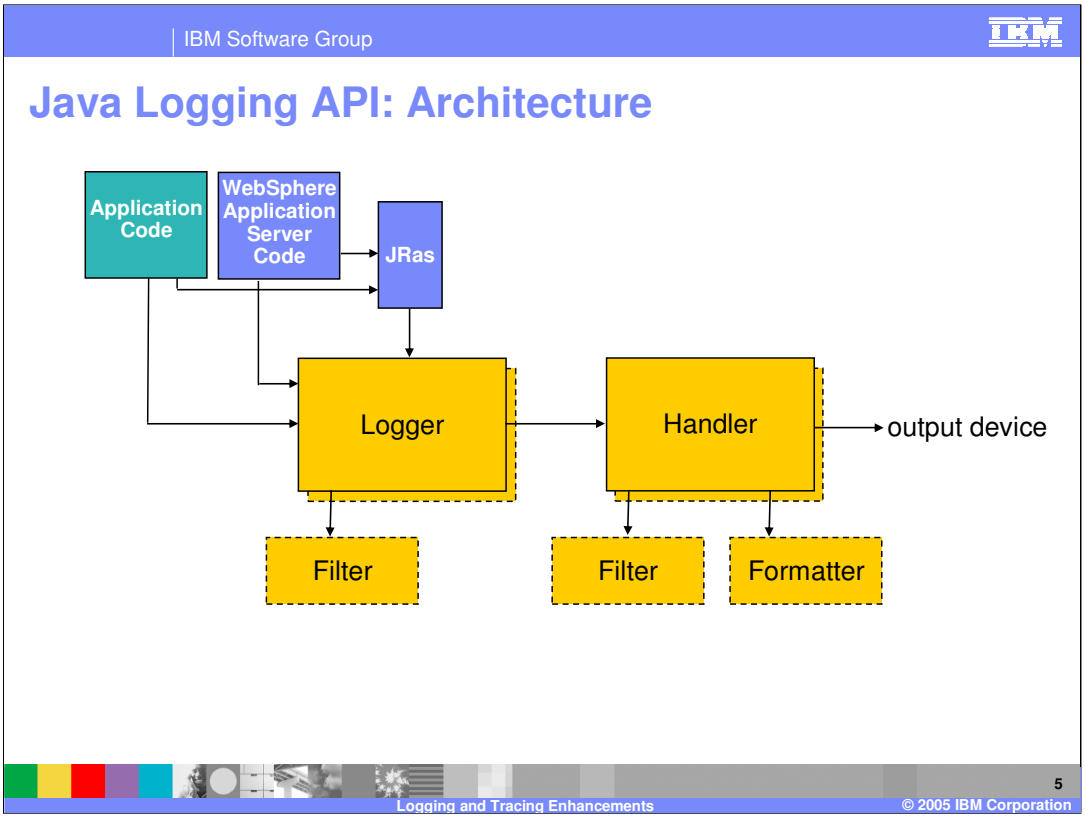
This section covers the Java logging API.



J2SE 1.4 includes a new package, called java.util.logging, introduced as JSR-047. Java logging provides a flexible and extensible logging API for your Java applications. Since it is a standard part of Java 1.4, your logging code will be portable to any runtime environment that supports J2SE 1.4.

The Java logging API is conceptually similar to JRas, the proprietary logging framework used by previous versions of WebSphere Application Server. While it is recommended that you use the Java logging API for any new code, JRas is still supported for compatibility with old applications that are migrated to V6. As you will see in the next slide, JRas has been integrated with the new logging framework so that JRas messages can be used in the same way as Java logging API messages.

Previous versions of WebSphere exposed an API called JRas. JSR-047 and JRas have similar functionality, but JSR-047 makes application logging portable to other compliant containers. Internally the JRas uses the Java Logging, which has become the core logging implementation and both write to the same log files.



This diagram shows the main elements of the Java logging architecture, and illustrates the flow of log data. Both application code and WebSphere Application Server code make use of Logger objects to put data onto the log stream, in the form of LogRecord objects. Logger objects can be associated with one or more Handler objects. Handlers represent output devices. For instance, one handler represents the service log, while another represents the StandardOut log. Java logging allows filters to be used to decide which messages get forwarded through the stream and which do not. For instance, you could exclude messages that contained a particular key using a filter. Formatters are used by Handlers to format log data for output. Localization could be implemented using a Formatter, for example.

JRas, the logging API used in previous versions of WebSphere Application Server, is integrated with the Java logging API, so that Loggers and Handlers can receive and process all messages, regardless of whether they were logged using JRas or Java logging. It is still possible for application code to utilize JRas as well, although it is suggested that new applications use the Java logging API.

Section

Configuring Logging and Tracing

This section covers the logging and tracing configuration using the Administrative Console.

Logging and Tracing Changes

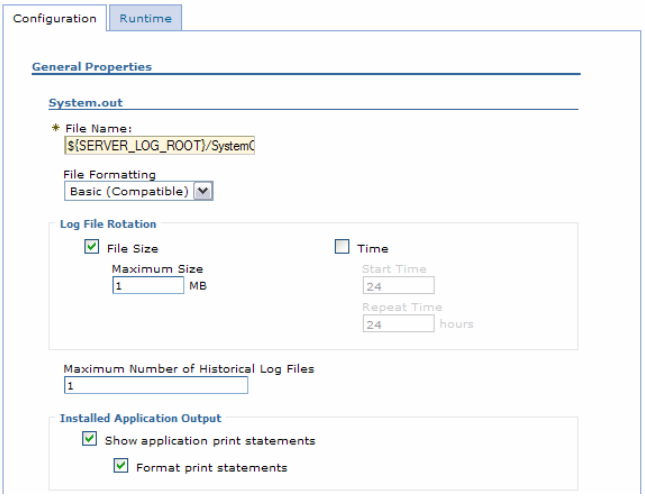
- Logging and tracing configuration panels are significantly changed
  - Related to the infrastructure changes made to implement Java logging
- Log Detail Level (formerly trace level) is now set on a separate panel
  - Stands alone from the Diagnostic Tracing panel because it affects both logging and tracing
  - Logging and tracing are the same from an infrastructure perspective
    - Different log and trace files correspond to different Java logging Handlers

As a result of the infrastructure changes made to support the Java logging framework, the logging and tracing configuration panels are significantly changed in V6. The most striking change you will see is that the Log Detail Level, formerly known as the trace level, is now set on its own panel, separate from the Diagnostic Tracing panel. This is because logging and tracing are the same from an infrastructure perspective in V6, and as a result the Log Detail Level affects both logging and tracing.

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 roll over based on time or file size
  - Number of historical log files is configurable

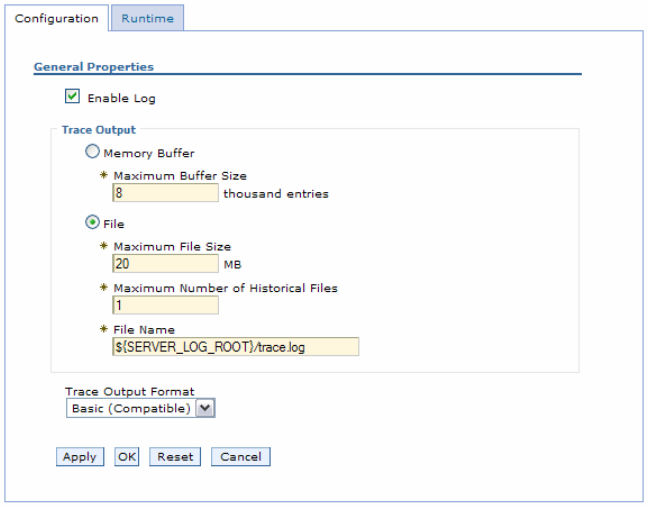


The JVM logs, System.out and System.err, can be configured by clicking “Logging and Tracing”, then “JVM Logs” from your Application Server’s main configuration page. The options available on this page are the same options that were available in V5, including size-based and time-based rollover of log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

Enabling and Configuring Trace File

Servers > Application Servers > *servername* > Diagnostic Trace Service

- "Enable Log" checkbox enables tracing
- Configurable output
  - Memory buffer or file
- Trace string entry moved to a separate panel



The Diagnostic Trace Service box looks mostly the same as it did in previous versions. The Configuration and Runtime tabs behave as they always have, with Configuration affecting the configuration repository and taking effect at the next startup, while Runtime takes effect immediately and can be optionally persisted to the server configuration.

You also still have the option to configure tracing to either a memory buffer or the file system.

The major change on this panel is the absence of a space to enter the trace string, now known as the Log Detail Level. This functionality has been moved to a separate panel.

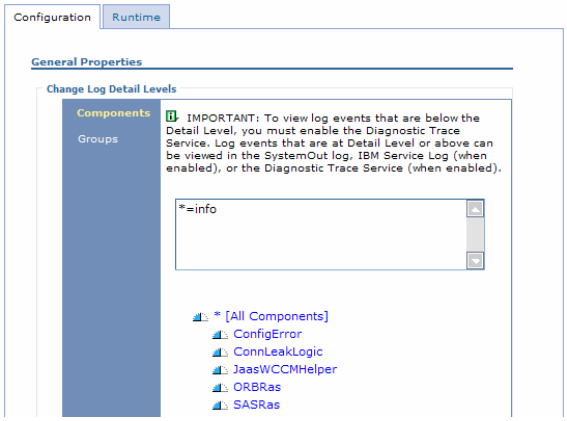
Log Detail Levels

v6 Levels	v5 Log Levels	v5 Trace Levels	v6 Description
Off	Off		Turn off logging and tracing
Fatal	Fatal		Task cannot continue and component cannot function.
Severe	Error		Task cannot continue but component can still function.
Warning	Warning		Potential error or impending error
Audit	Audit		Significant event affecting server state or resources
Info	Info		General information outlining overall task progress
Config			Configuration change or status
Detail			Info detailing subtask progress
Fine		Event	General trace
Finer		Entry / Exit	Detailed trace , method entry/exit trace
Finest		Debug	Most detailed trace
All		All=enabled	Log all events, including custom levels

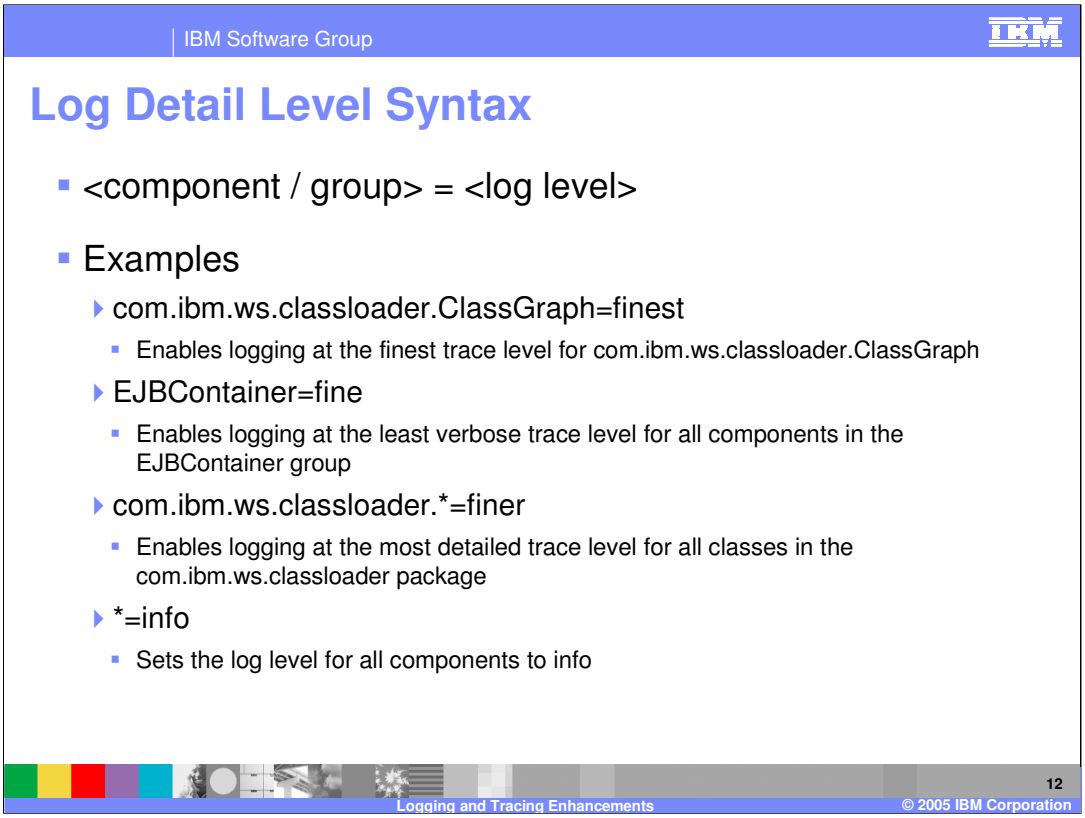
The available Log Detail Levels are different than the available levels in V5. These have been changed to include the levels defined by the Java logging specification. This table correlates the Log Detail Levels available in V6 with the log levels and trace levels available in V5. Remember that information at the levels “Fine”, “Finer”, “Finest”, and “All” will only be logged if you have enabled diagnostic tracing, regardless of the level set by the Log Detail Level string. It is recommended that you do not use the level “all”.

Setting the Log Detail Level

- *servername* > Logging and Tracing > Change Log Detail Level
- Log detail level affects tracing **and** regular logging
  - Setting levels below info reduces the amount of data in logs
- Trace-level output (fine, finer, finest) will not appear in trace file unless trace log is enabled
- Log string can be typed in or set using the graphical menu
- Default is *=info



You might remember setting trace strings in a pop-up window in v 5. The trace string, now called a Log Detail Level, has been moved to its own panel in v 6. This is because the Log Detail Level affects both logging and tracing. The default string, “*=info” enables regular logging, but disables tracing. Log levels can be set by typing in a string into the text field, or set graphically using the menu below the text field.



In addition to new Log Detail Levels, the strings used to specify them has also been simplified in V6, as defined by the Java logging specification. The new syntax is simply the component or group you want to trace, followed by an “equals” sign, and then the level of detail. For example, the string “com.ibm.ws.classloader.ClassGraph=finest” would enable logging at the finest trace level for the com.ibm.ws.classlooader.ClassGraph class. Wildcards are still supported, meaning you can use a wildcard to set the Log Detail Level for all classes in a package. You can also set the level for all components at the same time, as shown in the last example on this slide.

Log Detail Level Syntax (cont.)

- Separate multiple trace strings using colon (:)
  - For example, *=config:com.ibm.ejs.*=fine
- "*=<level>" sets level for all strings that are not explicitly set
  - Above string sets trace level fine for com.ibm.ejs.*, and log level config for all other components

To separate multiple trace strings, use a colon, just as you did in v 5. Any components that have not had a Log Detail Level explicitly set will use the level that has been specified for all components, using the star character.

Log Detail Level String Parsing

- Strings are parsed from left to right
  - If entries conflict, rightmost entry takes precedence
- If string does not start with “*=<level>”, “*=info” is prepended
  - Sets the default level for all strings not covered by the rest of the string
  - “com.ibm.ejs.*=fine” will result in “*=info:com.ibm.ejs.*=fine”

Log Detail Level strings are parsed from left to right, and if entries conflict, the rightmost entry takes precedence. If your Log Detail Level string does not start with the “star” character to set the default level for all components, “*=info” will be prepended to the string automatically. If you have set a default level later in the string, this will not affect you, since your level will be to the right and take precedence.

Compatibility with V5 Trace Strings

- v5-style trace strings are supported
  - Mapped to corresponding V6 log strings
  - Recommendation is to avoid using them
- “<level>=disabled” sets the log level to one level less verbose than <level>
- “<level>=enabled” sets the log level to <level>

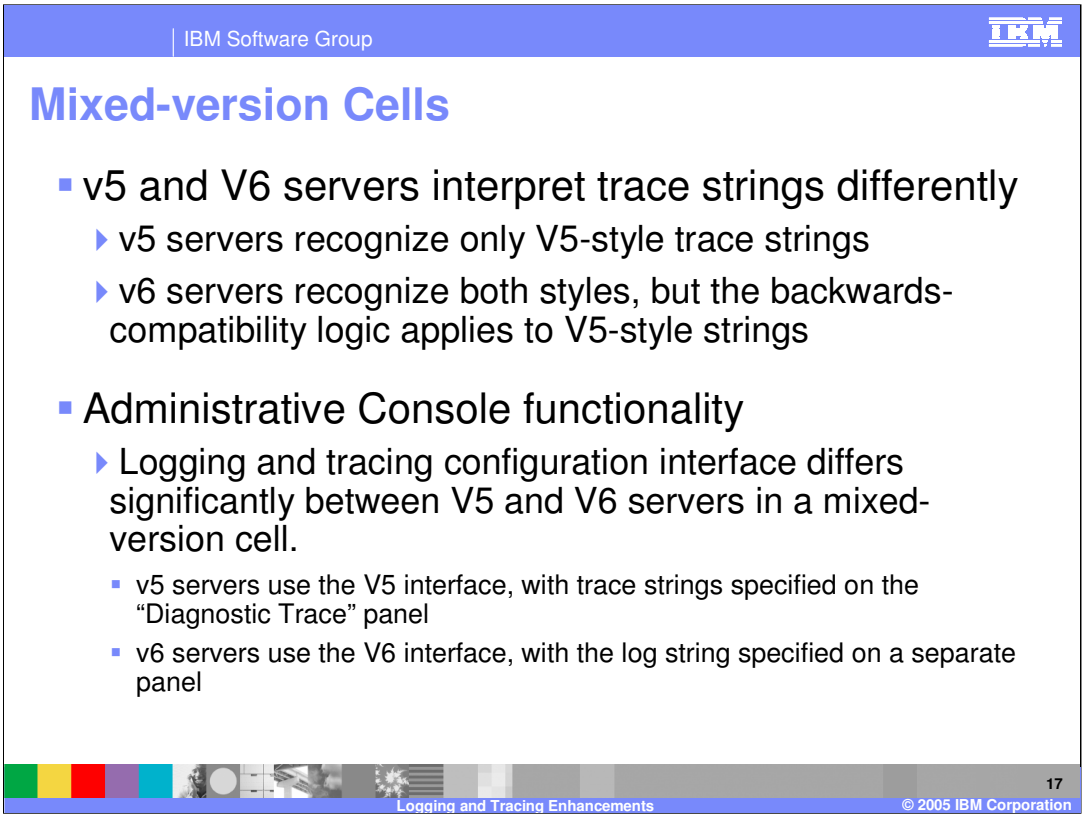
V5 style trace strings are supported in V6 for compatibility. If a V5-style string is used it will be mapped to the most similar V6 Log Detail Level based on the following logic: the “disabled” modifier sets the Log Detail Level to one step less verbose than the level specified in the string, while the “enabled” modifier sets the Log Detail Level to the level specified in the string. The recommendation is to use V6-style syntax.

v5 Trace Strings: Examples

v5 Trace String	Effect on v6	Why
com.ibm.ejs.ras.*=debug=enabled	com.ibm.ejs.ras.*=finest	*debug* is equivalent to *finest*, and *enabled* turns on the log level specified by the string
com.ibm.ejs.ras.*=debug=disabled	com.ibm.ejs.ras.*=finer	*debug* is equivalent to *finest*, and *disabled* sets the level to one level less verbose, which is *finer.*
*=all=enabled	*=all	*enabled* turns on the specified log level.
*=all=disabled	*=info	Turns off all tracing, but leaves logging enabled **This is an exception to the previous rules.**

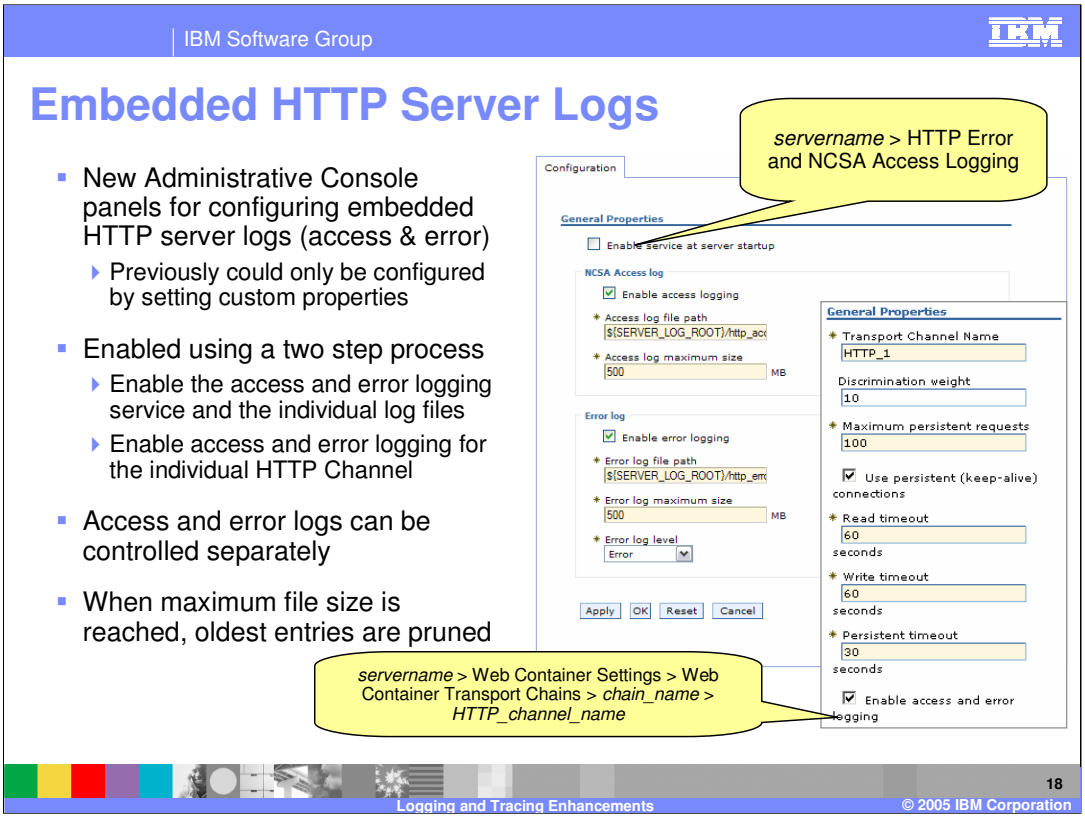
Reminder: It is recommended to use the V6-style log strings, not V5 style shown here

This table shows the result of using V5-style trace strings to illustrate how the mapping works. First the V6 Log Detail Level most similar to the specified level is found. Then if the “disabled” modifier is used, the level is set to one level less verbose. The exception to the previously specified processing rules is the string “*=all=disabled”, the default trace level in V5. If this string is used, it is equivalent to setting “*=info”, the default level in V6, which leaves logging enabled but disables tracing.



The compatibility with V5-style trace strings has some implications when using a mixed-version cell. First, V5 servers only understand V5-style trace strings, while V6 servers understand both styles. Second, even though the same V5-style string can be set on both versions, the string may cause slightly different behavior on each version in some cases.

Another thing to keep in mind is that the functionality built into the V6 Administrative Console exposes only the capabilities available to the particular server being configured. This means that while the interface for configuring logging and tracing has changed in V6, if you are using a V6 Deployment Manager to configure a V5 Application Server, you will actually see the old V5-style interface for configuring logging and tracing.



Although access and error logging have been available on the embedded HTTP transport since V5.0.2, they have lacked exposure in the Administrative Console and had to be configured using custom properties. In V6, each Application Server has a configuration panel for configuring HTTP access and error logging as shown here.

To enable the access or error log, you must check the “Enable service at server startup” checkbox, and also the checkbox for the specific log you want to enable. Notice there is no runtime tab for these logs. Once you have activated the logging service and the individual log files, you also need to enable access and error logging for each HTTP Channel that you want to write to the access or error logs. Logging will only begin once you have saved these changes to your configuration and restarted the Application Server.

Summary

- Logging and tracing has been changed internally to support Java logging (JSR-047)
  - Logging and tracing configuration is different as a result
- Embedded HTTP server logs can be configured from the console

In summary, this presentation has focused on the new logging and tracing functionality in WebSphere Application Server V6. These changes are a result of implementing the new Java logging specification that is part of J2SE 1.4. Logging and tracing configuration procedures have been changed significantly to support this new functionality.

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.