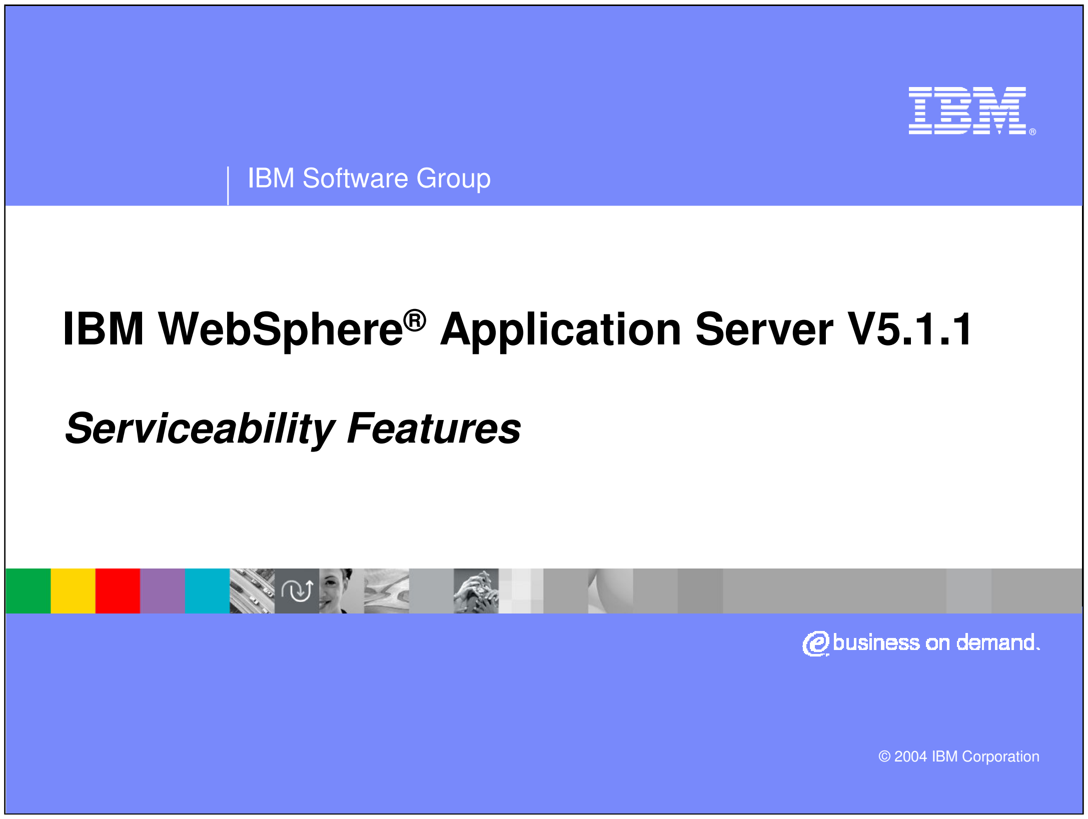
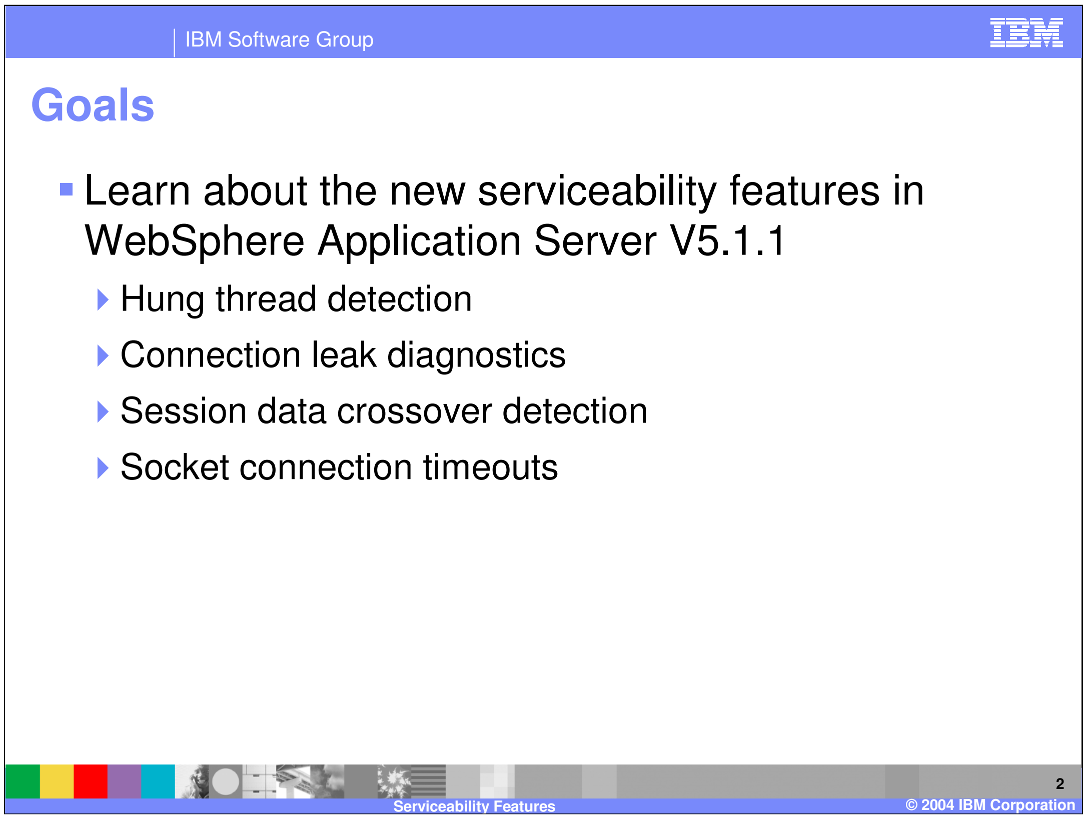


IBM Software Group

IBM WebSphere® Application Server V5.1.1

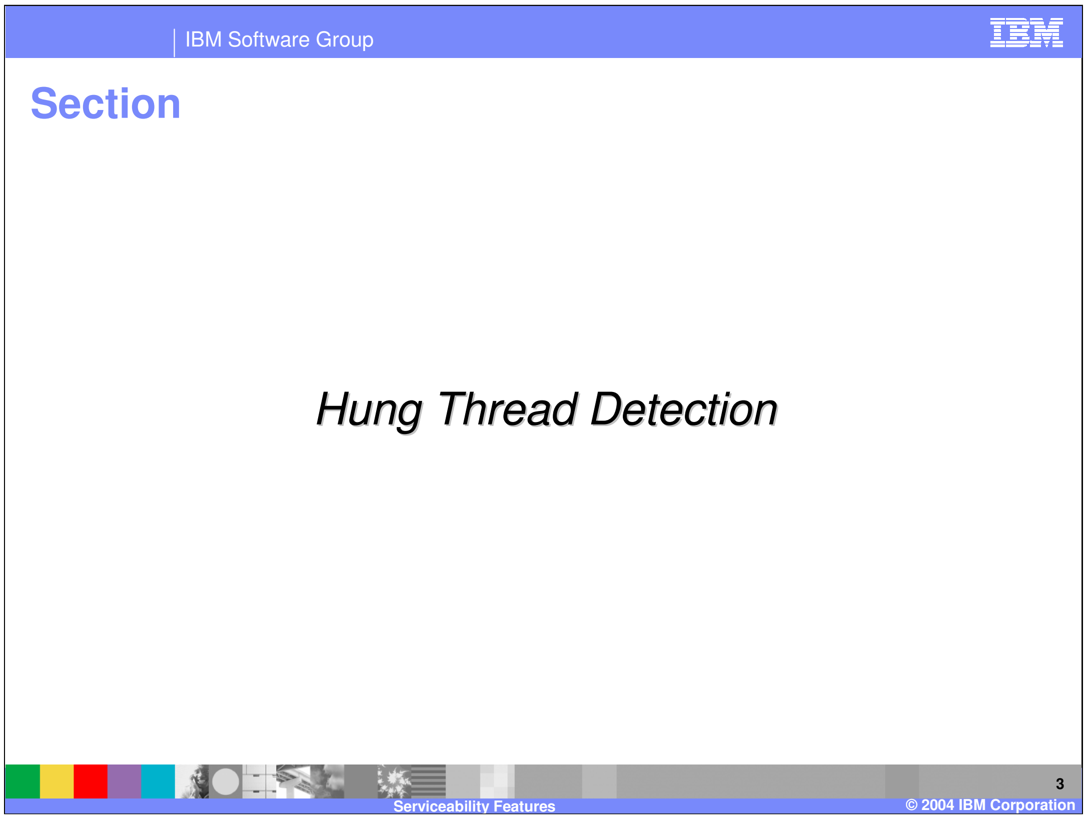
Serviceability Features

business on demand.



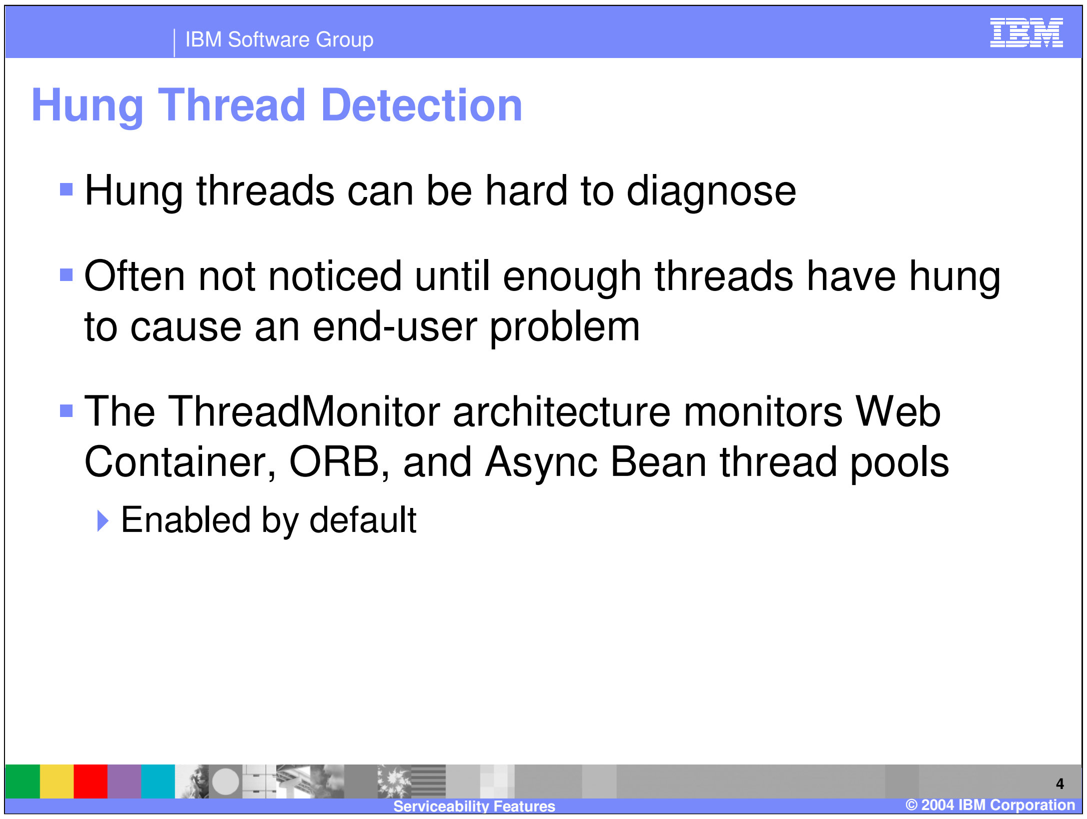
Goals

- Learn about the new serviceability features in WebSphere Application Server V5.1.1
  - Hung thread detection
  - Connection leak diagnostics
  - Session data crossover detection
  - Socket connection timeouts



Section

Hung Thread Detection

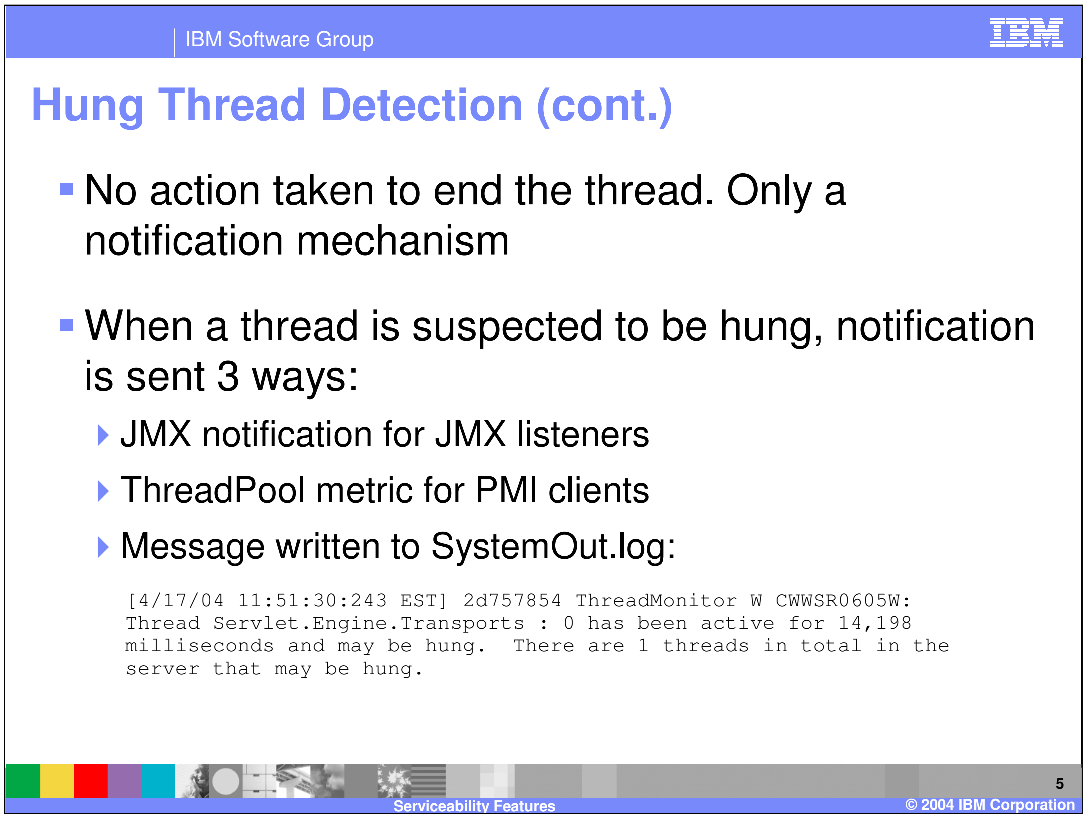


Hung Thread Detection

- Hung threads can be hard to diagnose
- Often not noticed until enough threads have hung to cause an end-user problem
- The ThreadMonitor architecture monitors Web Container, ORB, and Async Bean thread pools
  - Enabled by default

Application threads can hang for a number of reasons, including infinite loops or deadlocks.

As of version 5.1.1, there is a component known as the ThreadMonitor that monitors the Web Container, ORB, and Async Bean thread pools for hung threads.



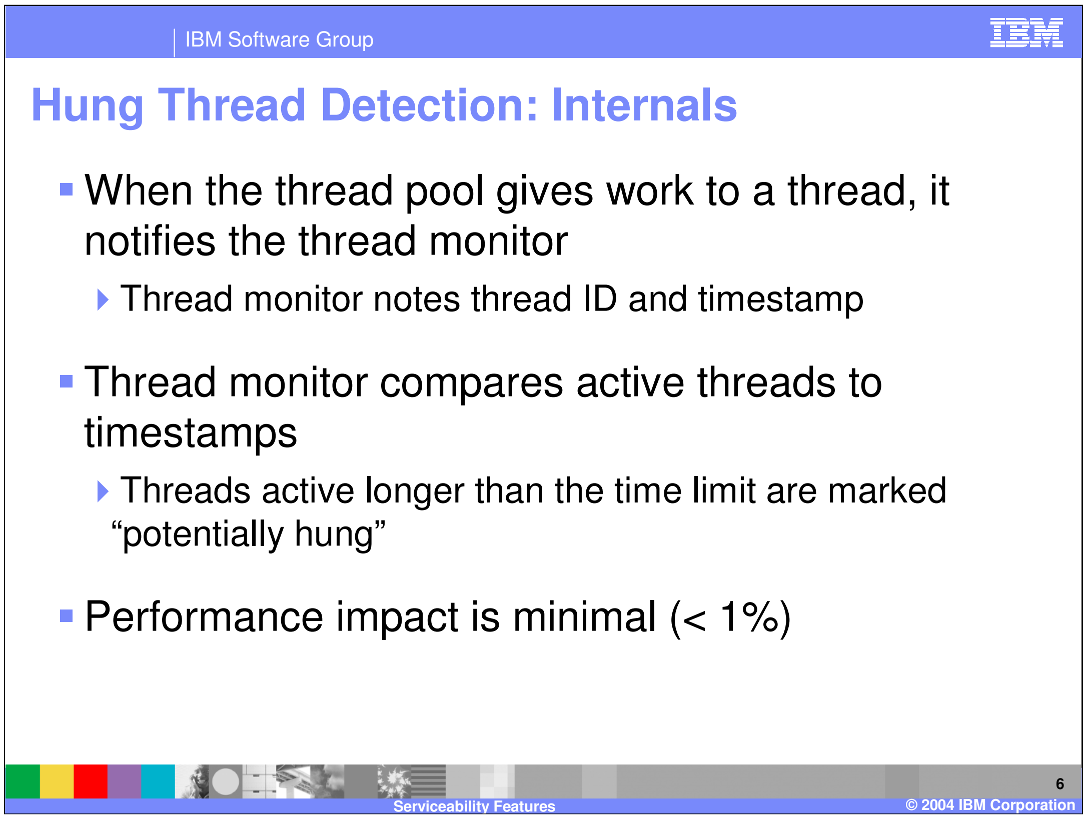
Hung Thread Detection (cont.)

- No action taken to end the thread. Only a notification mechanism
- When a thread is suspected to be hung, notification is sent 3 ways:
  - JMX notification for JMX listeners
  - ThreadPool metric for PMI clients
  - Message written to SystemOut.log:

```
[4/17/04 11:51:30:243 EST] 2d757854 ThreadMonitor W CWWSR0605W:
Thread Servlet.Engine.Transports : 0 has been active for 14,198
milliseconds and may be hung.  There are 1 threads in total in the
server that may be hung.
```

The thread monitor does not try to deal with the hung threads, it just issues notifications, so that the administrator or developer can deal with the issues.

When a hung thread is detected, three notifications are sent: a JMX notification for JMX listeners, PMI Thread Pool data is updated for tools like the Tivoli Performance Viewer, and a message is written to the SystemOut log.



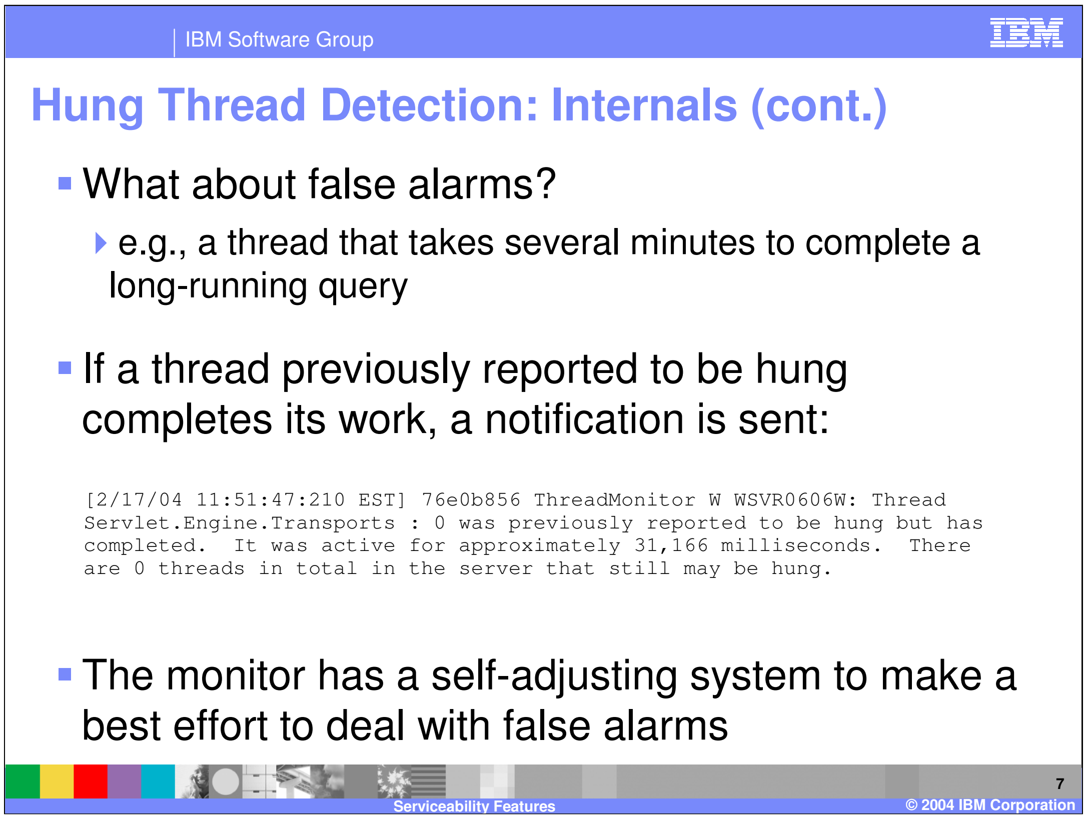
Hung Thread Detection: Internals

- When the thread pool gives work to a thread, it notifies the thread monitor
  - Thread monitor notes thread ID and timestamp
- Thread monitor compares active threads to timestamps
  - Threads active longer than the time limit are marked "potentially hung"
- Performance impact is minimal (< 1%)

When the thread pool issues work to a thread, it sends a notification to the thread monitor, which notes the thread ID and the time in a list.

At user-configurable intervals, the thread monitor looks at the active threads, and compares them to the list, to determine how long each thread has been active. If a thread has been active longer than the user-specified threshold, the thread is marked as "potentially hung", and the notifications are sent as discussed on the previous slide.

The performance impact of this monitoring is minimal. Less than 1%.



Hung Thread Detection: Internals (cont.)

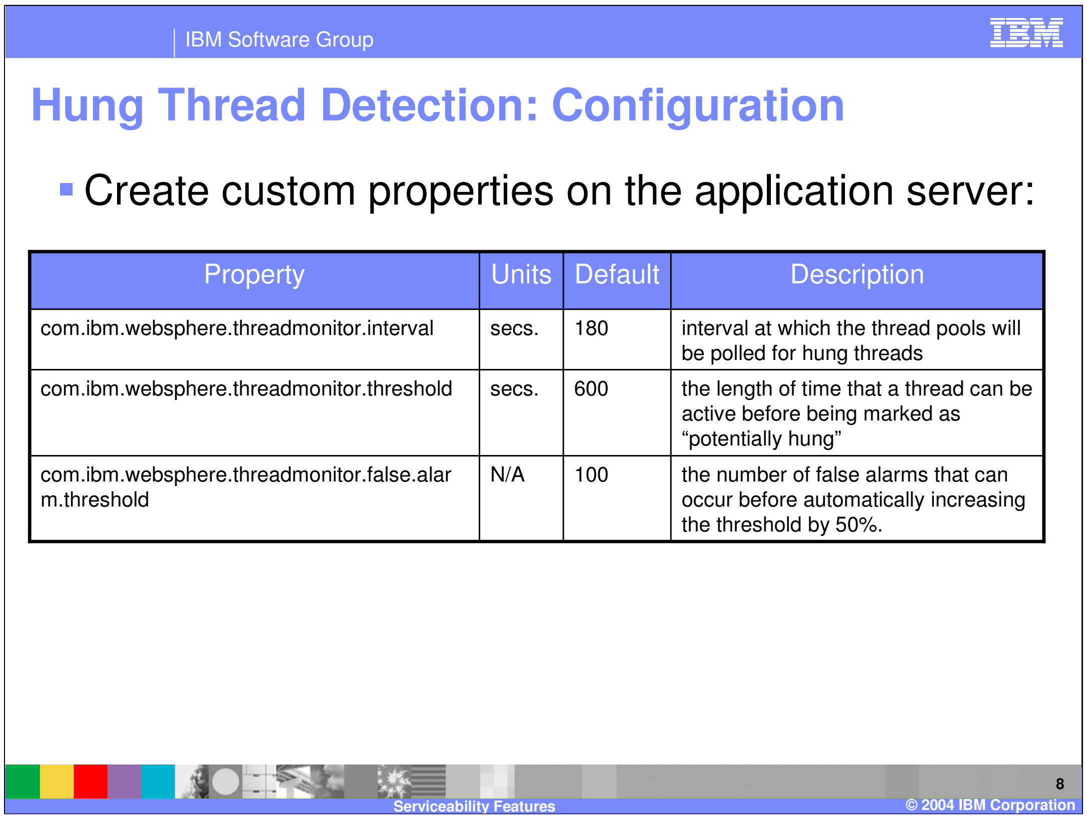
- What about false alarms?
  - e.g., a thread that takes several minutes to complete a long-running query
- If a thread previously reported to be hung completes its work, a notification is sent:

```
[2/17/04 11:51:47:210 EST] 76e0b856 ThreadMonitor W WSVR0606W: Thread
Servlet.Engine.Transports : 0 was previously reported to be hung but has
completed.  It was active for approximately 31,166 milliseconds.  There
are 0 threads in total in the server that still may be hung.
```

- The monitor has a self-adjusting system to make a best effort to deal with false alarms

It's possible that a thread could actually be running for longer than the specified threshold for legitimate reasons. For example, a thread could be executing a large database query that takes several minutes to return.

The thread monitor is built to recognize false alarms and adjust itself automatically. When a thread that was previously marked as "potentially hung" completes its work and exits, a notification is sent. After a certain number of false alarms, the threshold is automatically increased by 50% to account for these long-running threads. The idea is that if there are several threads that are routinely active for 20 minutes, the threshold will eventually adjust itself to be higher than 20 minutes, so as to not mark those threads as hung.



Hung Thread Detection: Configuration

- Create custom properties on the application server:

Property	Units	Default	Description
com.ibm.websphere.threadmonitor.interval	secs.	180	interval at which the thread pools will be polled for hung threads
com.ibm.websphere.threadmonitor.threshold	secs.	600	the length of time that a thread can be active before being marked as “potentially hung”
com.ibm.websphere.threadmonitor.false.alarm.threshold	N/A	100	the number of false alarms that can occur before automatically increasing the threshold by 50%.

The hang detection policy can be configured by creating custom properties for the application server.

com.ibm.threadmonitor.interval is the interval at which the thread pools will be polled for hung threads (in seconds). It defaults to 180 seconds, which is 3 minutes.

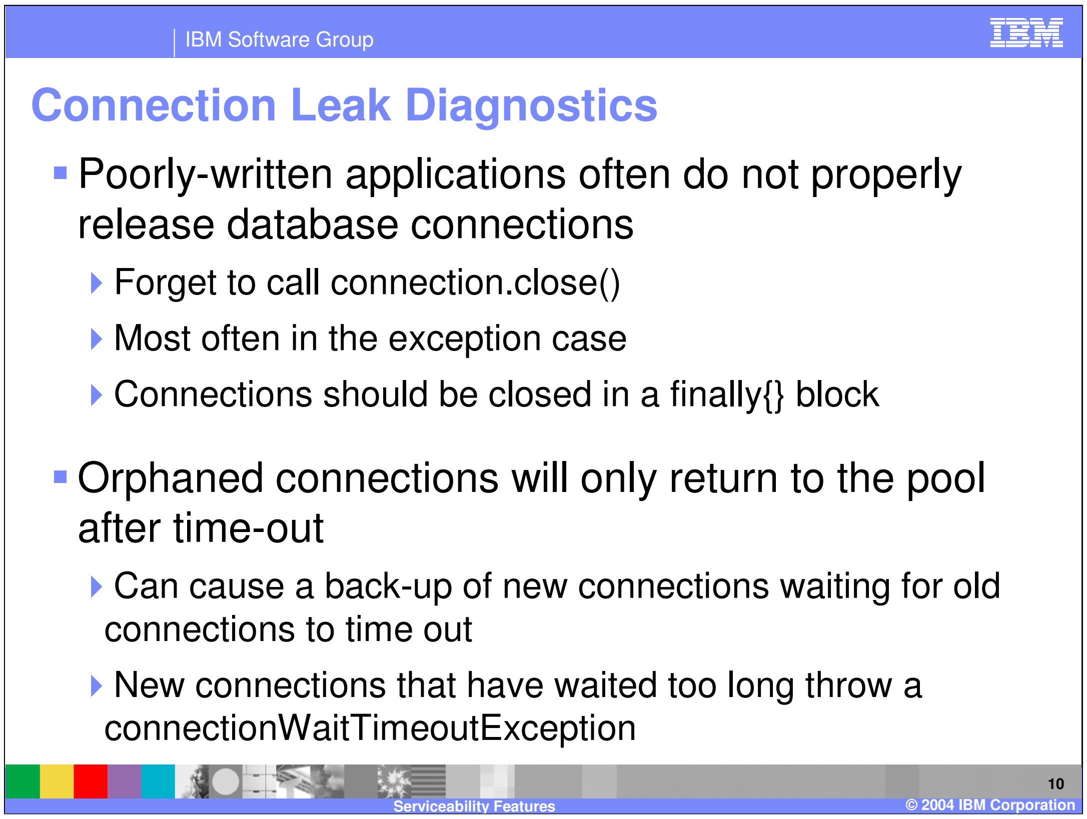
com.ibm.websphere.threadmonitor.threshold is the length of time that a thread can be active before being marked as “potentially hung”. The default value is ten minutes.

com.ibm.websphere.threadmonitor.false.alarm.threshold is the number of false alarms that can occur before automatically increasing the threshold by 50%. The default value is 100. Automatic adjustment can be disabled altogether by setting this property to zero.

The application server must be restarted for these changes to take effect. To adjust the hang detection policy on the fly, use wsadmin. Refer to the Information Center for instructions.

Section

Connection Leak Diagnostics



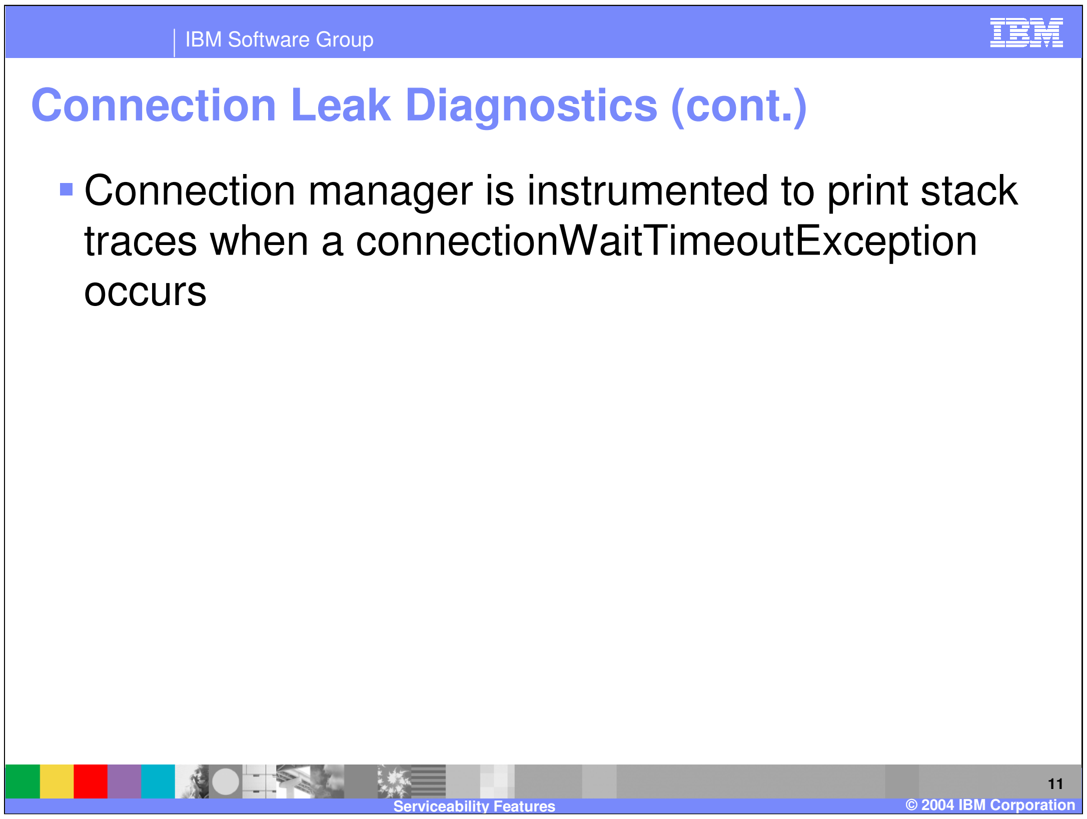
Connection Leak Diagnostics

- Poorly-written applications often do not properly release database connections
  - Forget to call connection.close()
  - Most often in the exception case
  - Connections should be closed in a finally{} block
- Orphaned connections will only return to the pool after time-out
  - Can cause a back-up of new connections waiting for old connections to time out
  - New connections that have waited too long throw a connectionWaitTimeoutException

Applications can suffer from performance problems and even appear to “hang” if they do not close their connections properly. This is most often caused by developers not properly using the connection.close() method. To ensure that connections will be closed properly, they should be closed in a finally{} block.

WebSphere is smart enough to eventually time-out orphaned connections and return them to the pool, but for an application that makes frequent use of database connections, this might not be enough. New connections can get queued up waiting for the database while old connections are waiting to be timed out. This can bring the application grinding to a halt, and you can see connectionWaitExceptons.

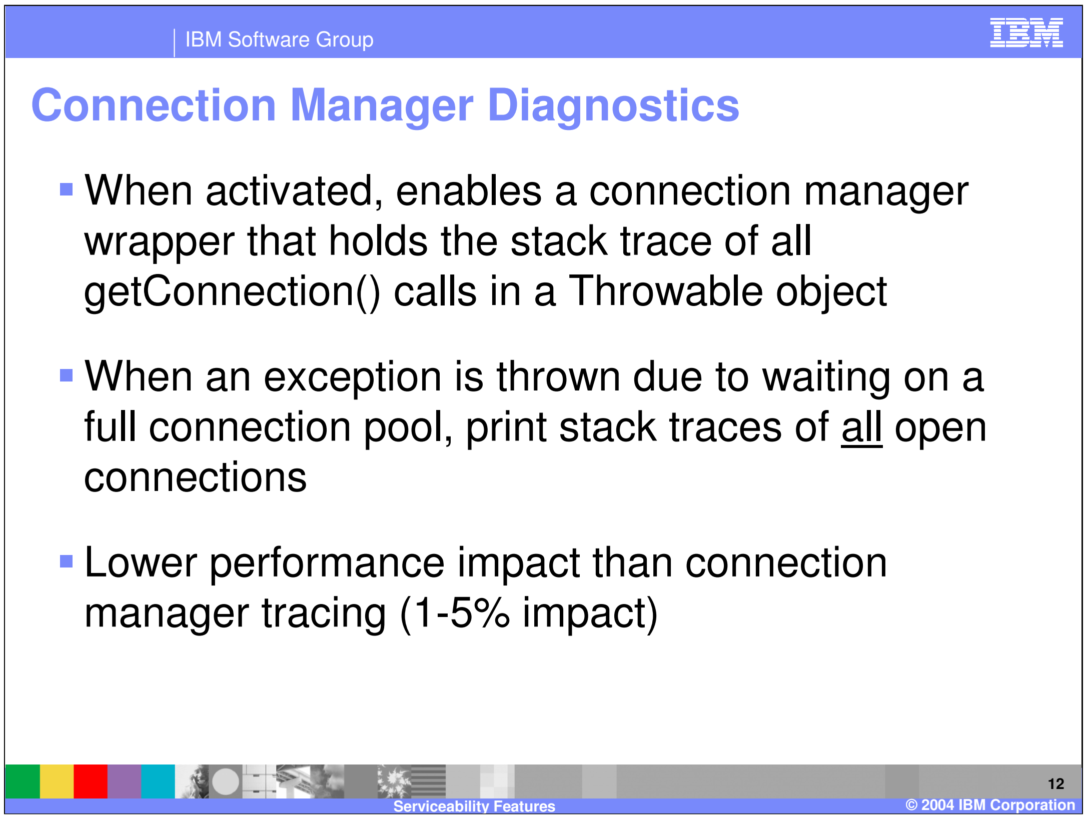
Connection leaks have traditionally been hard to diagnose because the error messages do not usually provide specific enough information about the source of the problem. Usually a source code review is needed to find points in the code where connections are not properly closed.



Connection Leak Diagnostics (cont.)

- Connection manager is instrumented to print stack traces when a connectionWaitTimeoutException occurs

The connection manager has new instrumentation that can hold stack traces for all code that calls getConnection().

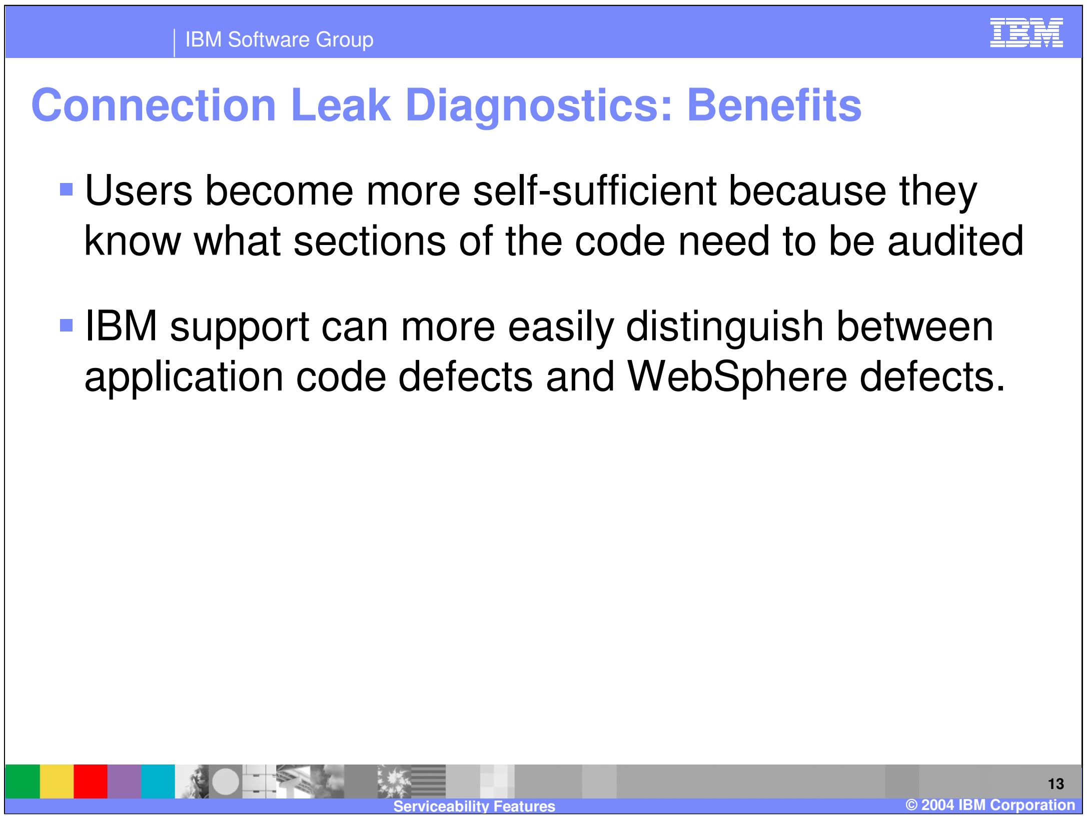


Connection Manager Diagnostics

- When activated, enables a connection manager wrapper that holds the stack trace of all getConnection() calls in a Throwable object
- When an exception is thrown due to waiting on a full connection pool, print stack traces of all open connections
- Lower performance impact than connection manager tracing (1-5% impact)

When a thread times out waiting on a connection from a full connection pool, it will throw a connectionWaitTimeoutException.

When this exception is thrown, the wrapper will print out the stack traces for every open connection.

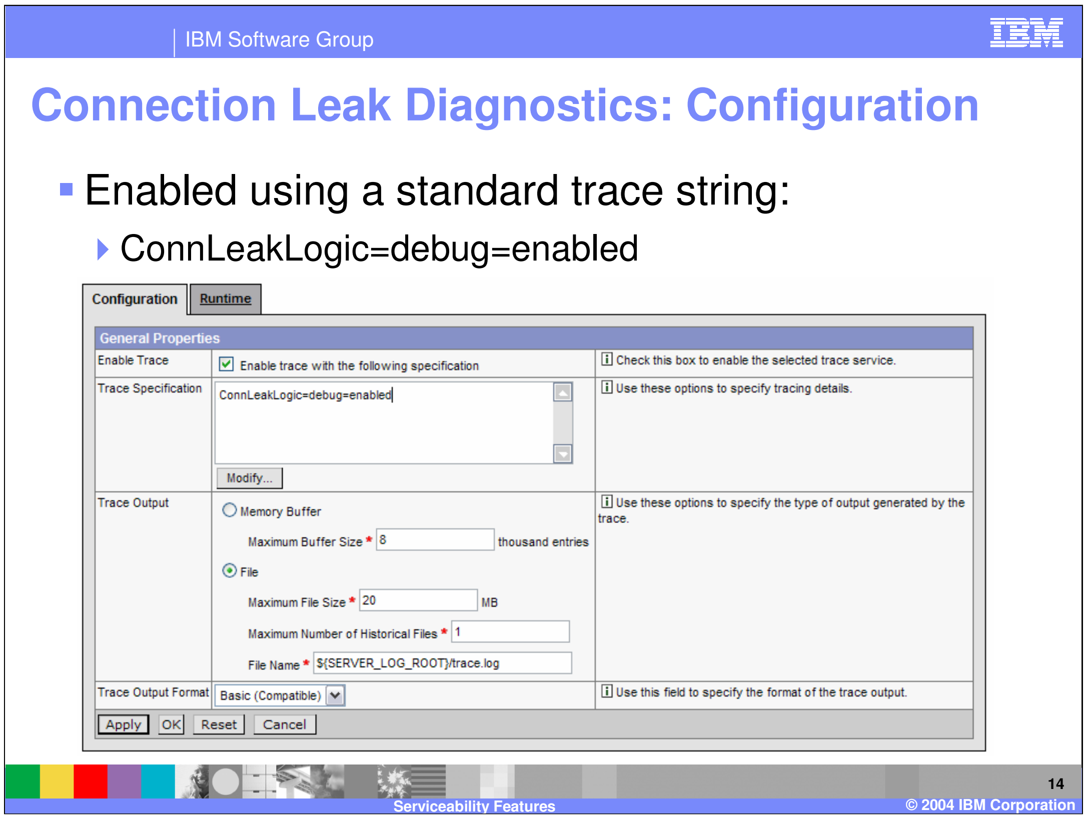


Connection Leak Diagnostics: Benefits

- Users become more self-sufficient because they know what sections of the code need to be audited
- IBM support can more easily distinguish between application code defects and WebSphere defects.

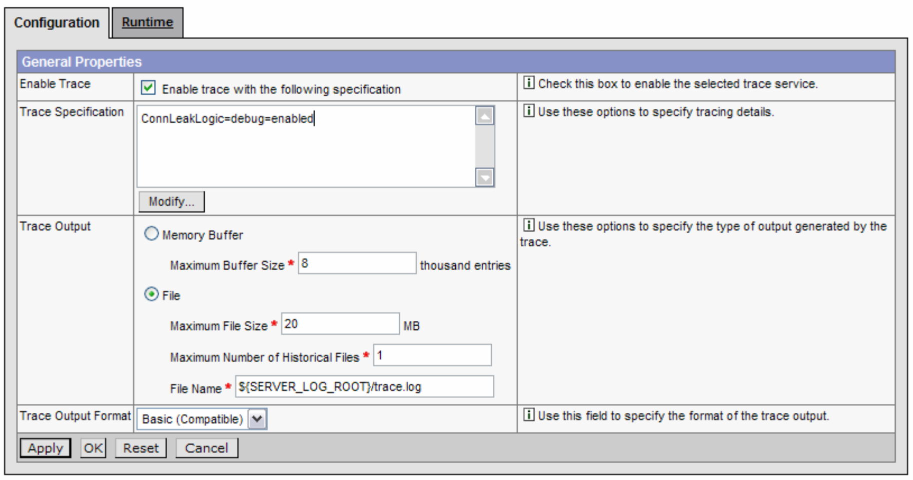
This feature is useful because it shows you the call stacks for all open connections at the time of the exception. This enables you to significantly narrow your search area when you look at the application's source code to try and find the responsible code.

It will also be helpful for IBM support, because it will help distinguish between application defects and WebSphere defects.



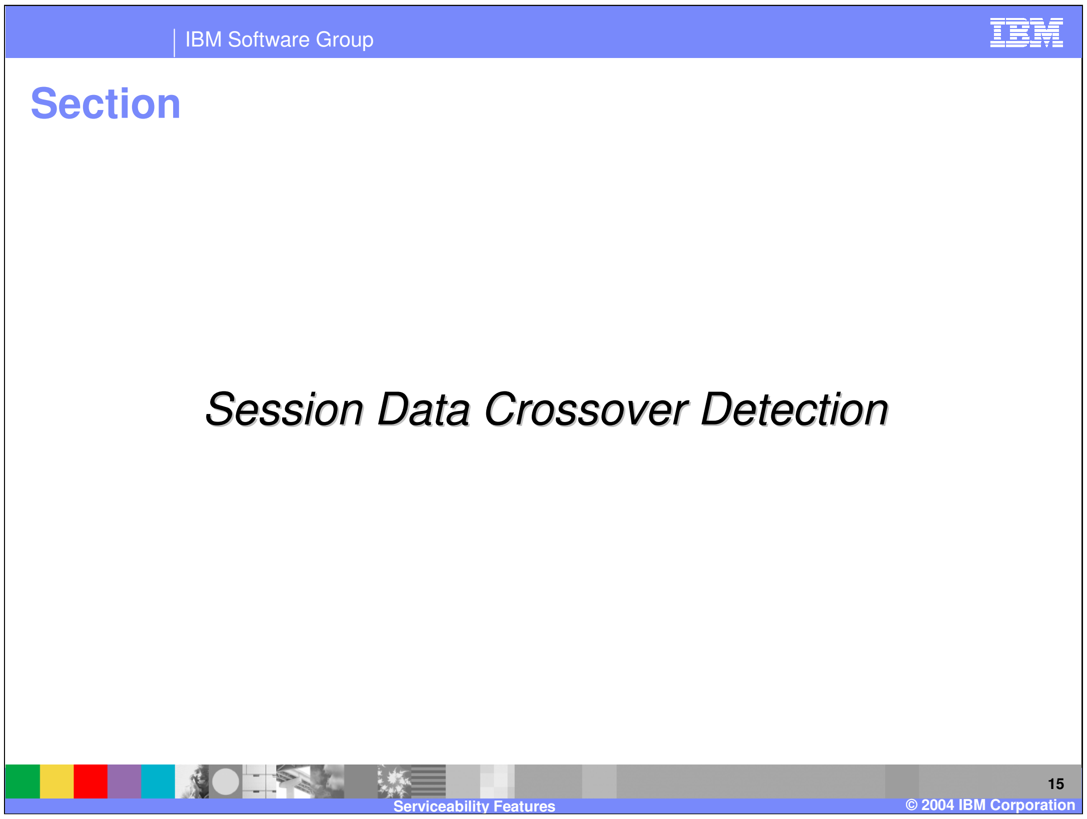
Connection Leak Diagnostics: Configuration

- Enabled using a standard trace string:
  - ConnLeakLogic=debug=enabled



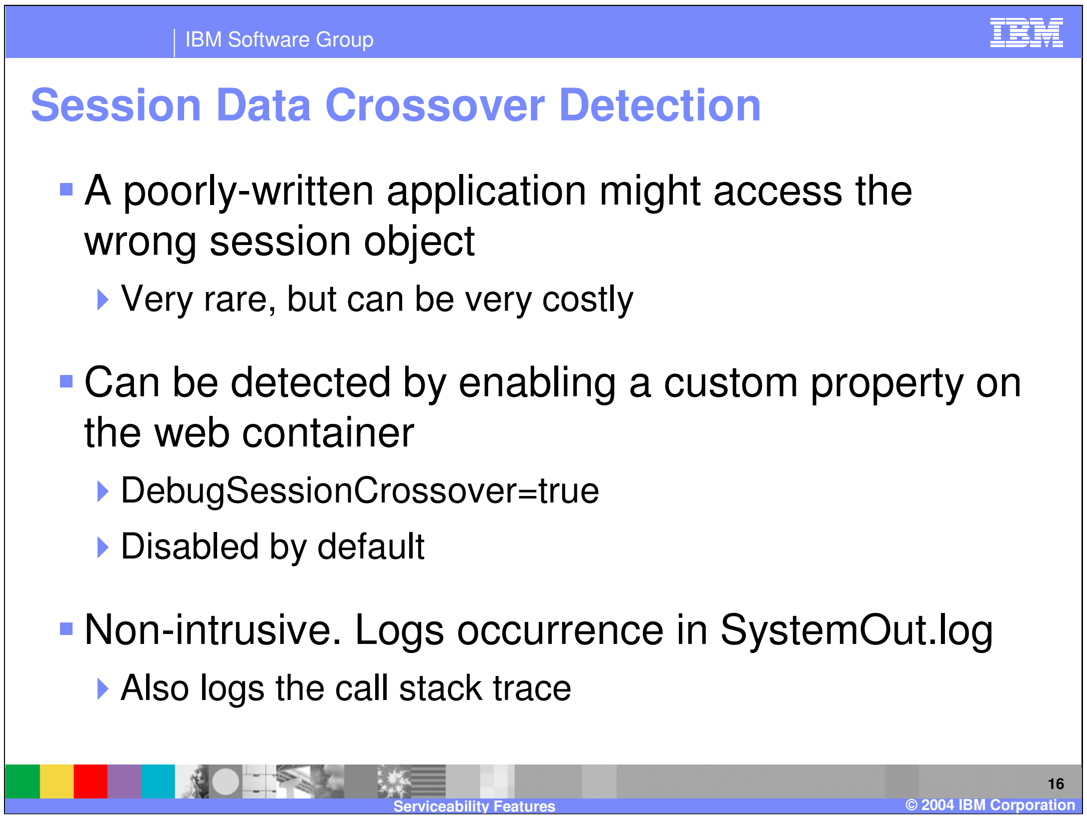
Enable Trace	[✓] Enable trace with the following specification	Check this box to enable the selected trace service.
Trace Specification	ConnLeakLogic=debug=enabled	Use these options to specify tracing details.
Trace Output	Memory Buffer Maximum Buffer Size * 8 thousand entries [✓ File] Maximum File Size * 20 MB Maximum Number of Historical Files * 1 File Name * ${SERVER_LOG_ROOT}/trace.log	Use these options to specify the type of output generated by the trace.
Trace Output Format	Basic (Compatible)	Use this field to specify the format of the trace output.

Connection leak diagnostics are enabled using the trace string shown here.



Section

Session Data Crossover Detection



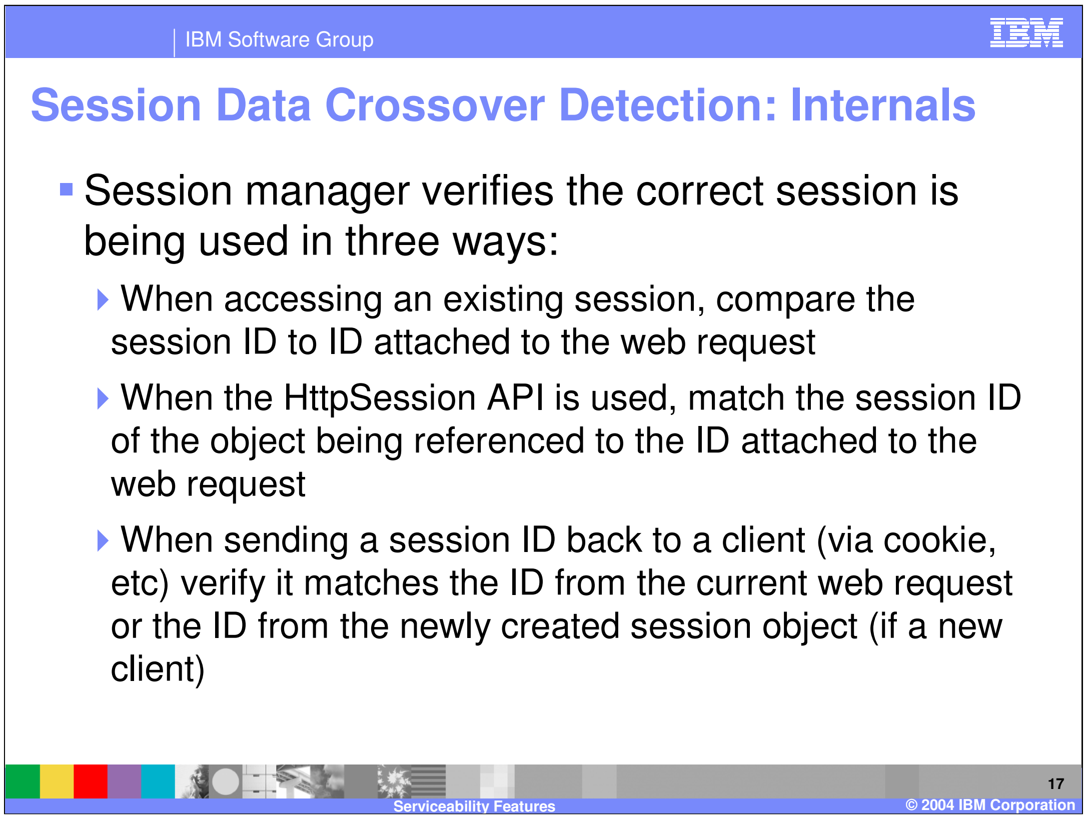
Session Data Crossover Detection

- A poorly-written application might access the wrong session object
  - Very rare, but can be very costly
- Can be detected by enabling a custom property on the web container
  - DebugSessionCrossover=true
  - Disabled by default
- Non-intrusive. Logs occurrence in SystemOut.log
  - Also logs the call stack trace

It is possible for a poorly-written application to access the wrong session object. When this occurs, it is known as "session data crossover". This is a rare problem, because it requires an application to modify the session ID that was sent by the client, but if it does happen, the result can be very costly.

WebSphere now includes logic to check for session data crossover. It is disabled by default, because enabling it incurs a roughly 3% performance penalty. It can be enabled by setting a custom property on the web container.

This detection mechanism is non-intrusive. No action is taken to stop the misuse of the session. The occurrence will be logged in StandardOut.log, and the stack trace of the offending function will also be logged.



Session Data Crossover Detection: Internals

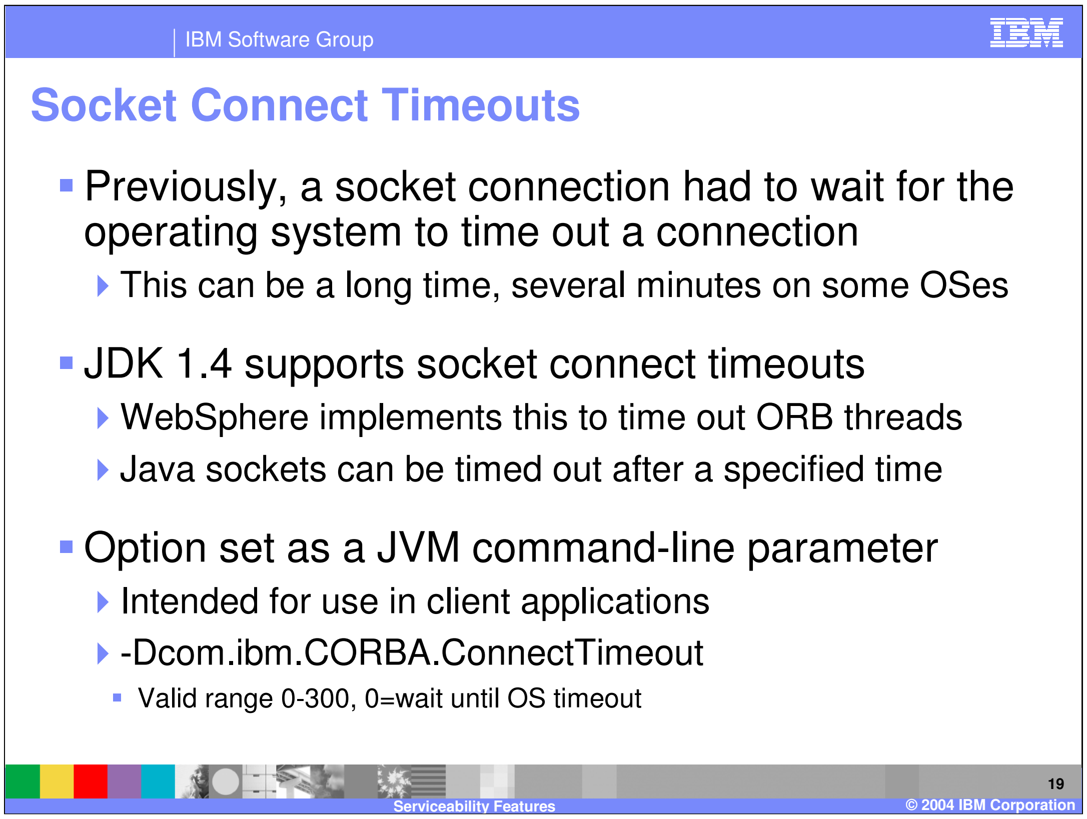
- Session manager verifies the correct session is being used in three ways:
  - When accessing an existing session, compare the session ID to ID attached to the web request
  - When the HttpSession API is used, match the session ID of the object being referenced to the ID attached to the web request
  - When sending a session ID back to a client (via cookie, etc) verify it matches the ID from the current web request or the ID from the newly created session object (if a new client)

The first and second cases are similar. The first being when a session is accessed using getSession(), and the second being when any method from the HttpSession package is called. The third case is when the session ID is being sent in the opposite direction, back to the client.

If the session IDs do not match in any of these three situations, the session manager will log a message to StandardOut.log.

Section

Socket Connect Timeouts



IBM Software Group

Socket Connect Timeouts

- Previously, a socket connection had to wait for the operating system to time out a connection
  - This can be a long time, several minutes on some OSes
- JDK 1.4 supports socket connect timeouts
  - WebSphere implements this to time out ORB threads
  - Java sockets can be timed out after a specified time
- Option set as a JVM command-line parameter
  - Intended for use in client applications
  - -Dcom.ibm.CORBA.ConnectTimeout
    - Valid range 0-300, 0=wait until OS timeout

19

Serviceability Features

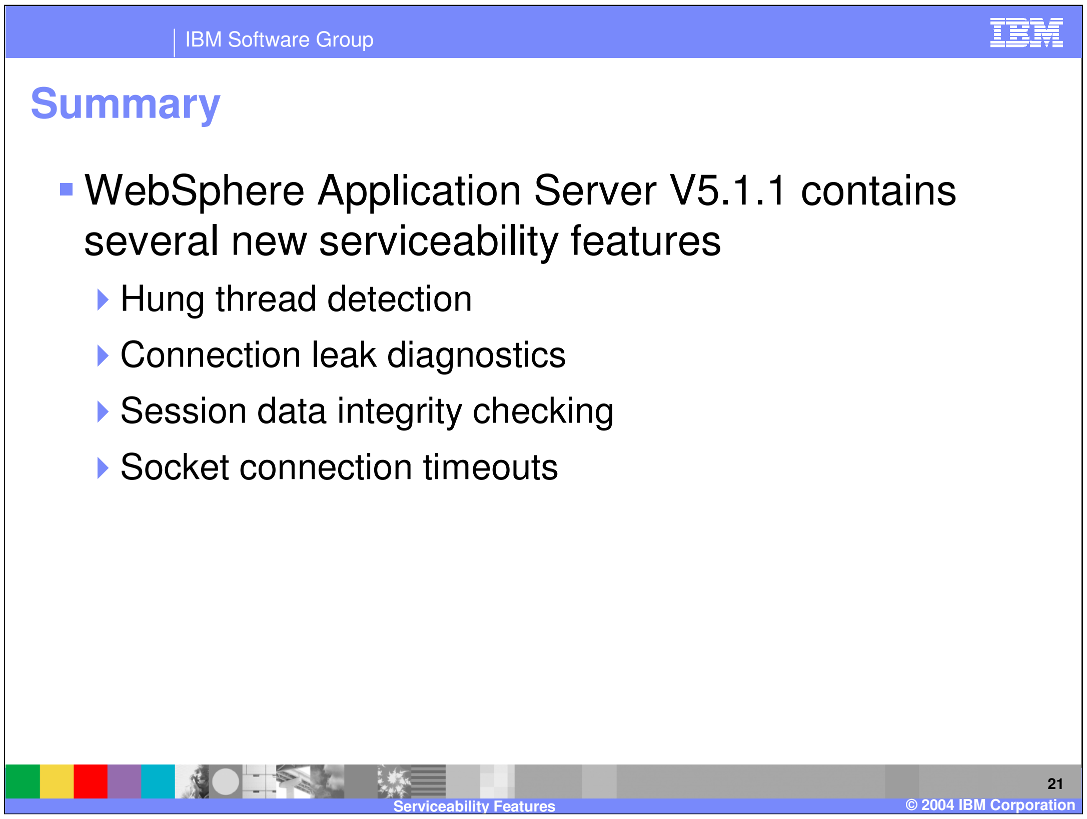
© 2004 IBM Corporation

This timeout is handled by the JVM. It does not modify your OS level timeout.

This also means that if your OS timeout is lower than this value, it will still time out when the OS level timeout expires.

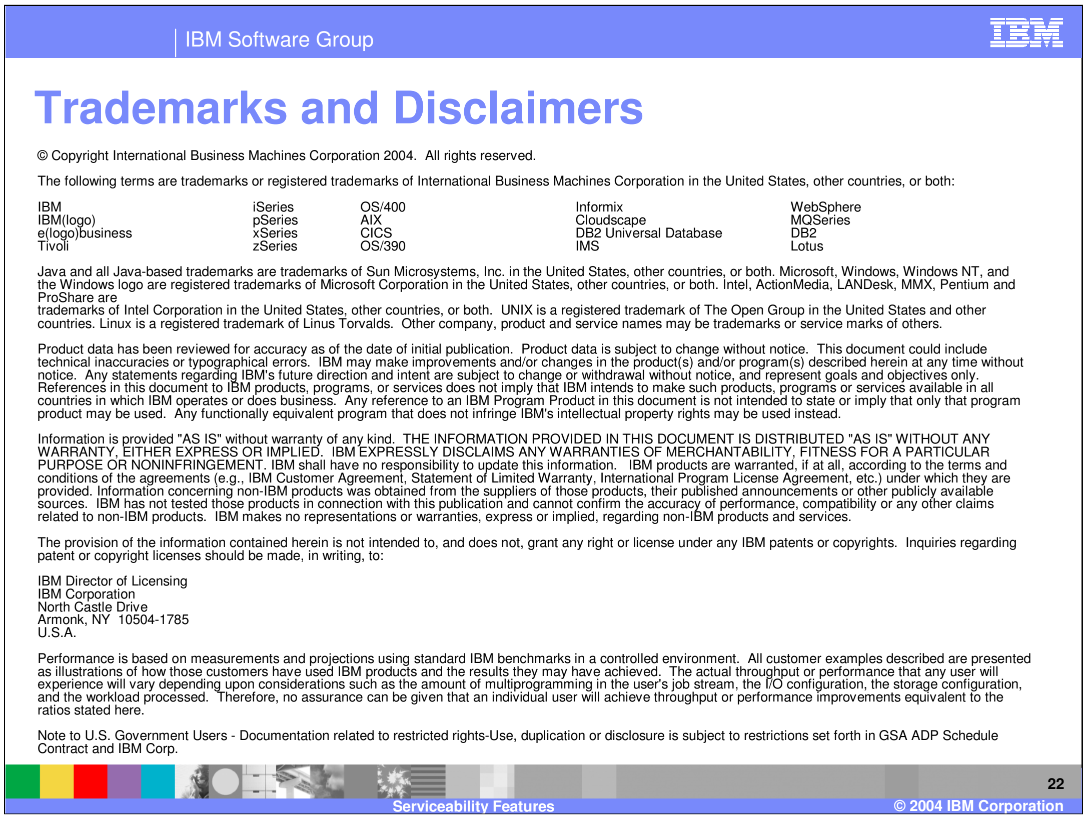
Section

Summary



Summary

- WebSphere Application Server V5.1.1 contains several new serviceability features
  - Hung thread detection
  - Connection leak diagnostics
  - Session data integrity checking
  - Socket connection timeouts



Trademarks and Disclaimers