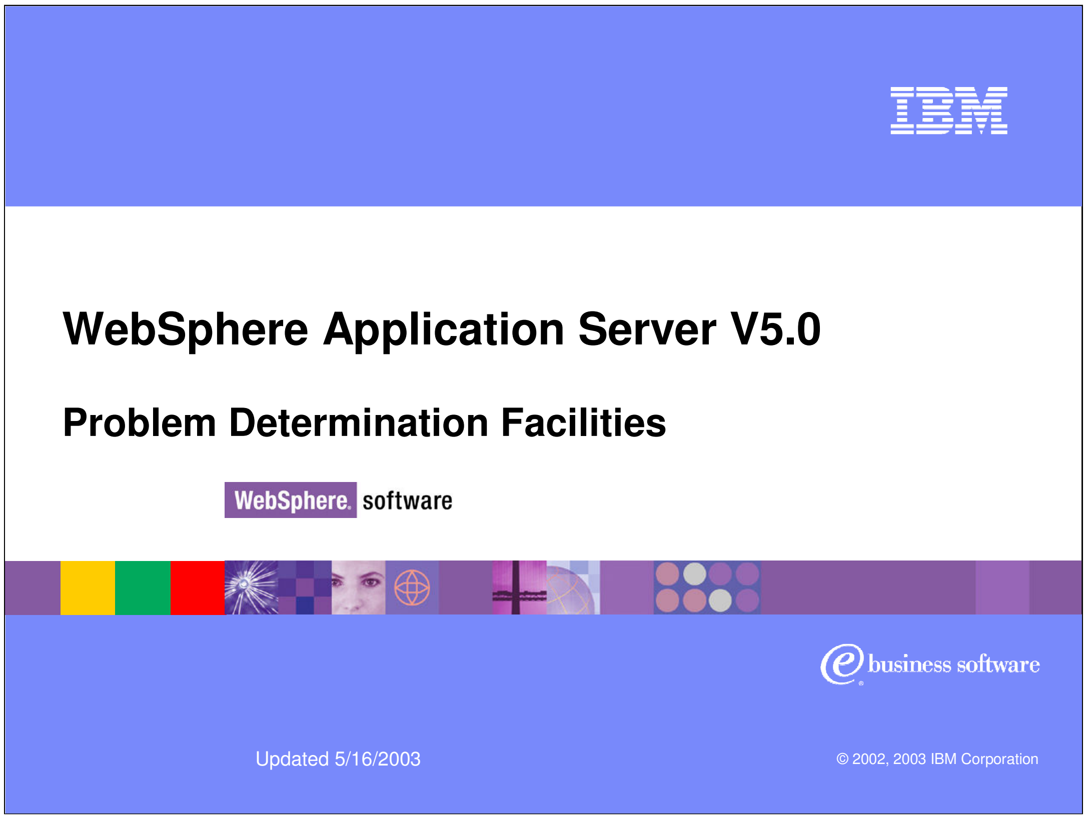
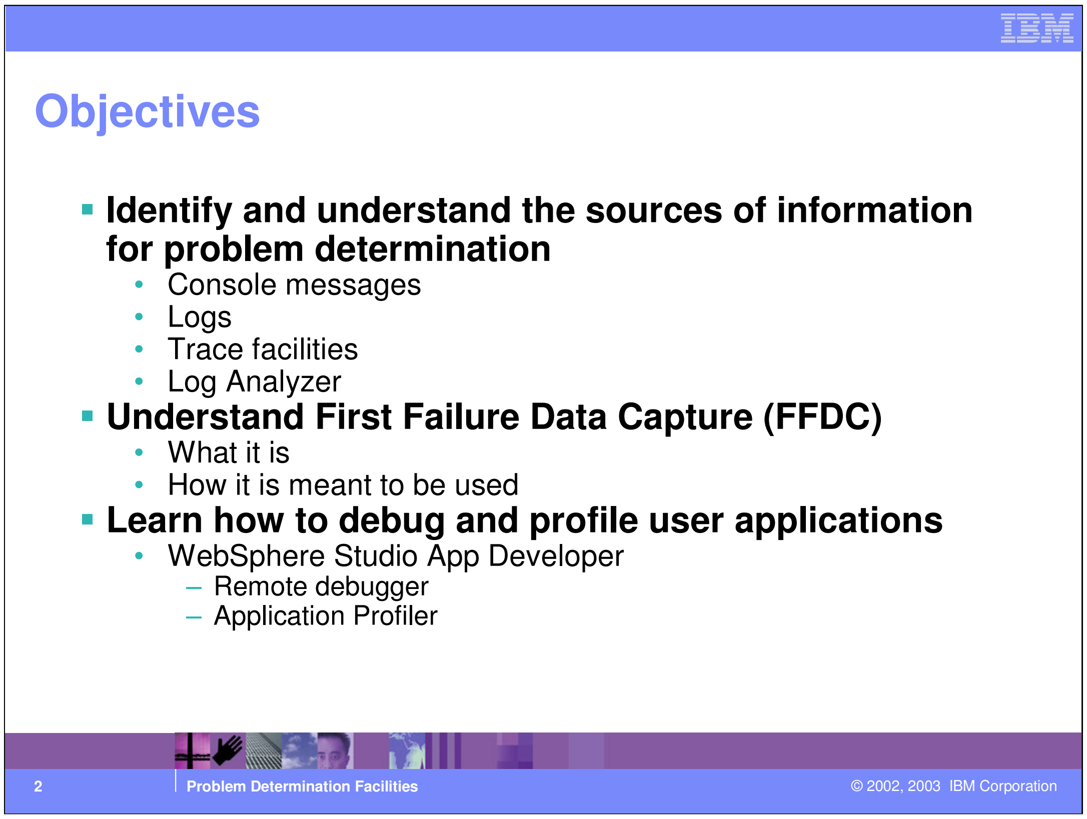


WebSphere Application Server V5.0

Problem Determination Facilities

Updated 5/16/2003



Objectives

- **Identify and understand the sources of information for problem determination**
  - Console messages
  - Logs
  - Trace facilities
  - Log Analyzer
- **Understand First Failure Data Capture (FFDC)**
  - What it is
  - How it is meant to be used
- **Learn how to debug and profile user applications**
  - WebSphere Studio App Developer
    - Remote debugger
    - Application Profiler

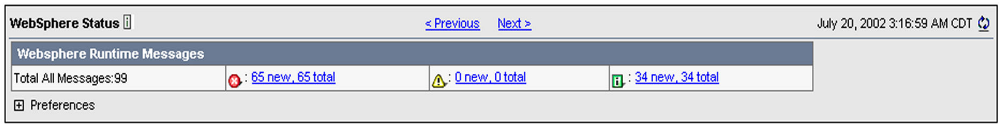
2 | Problem Determination Facilities © 2002, 2003 IBM Corporation

Migration involves moving the WebSphere runtime and the customer configuration settings from version 3.5.x or 4.0.x to version 5.0

Console Messages

Console Messages

- **Two types of messages**
  - Runtime

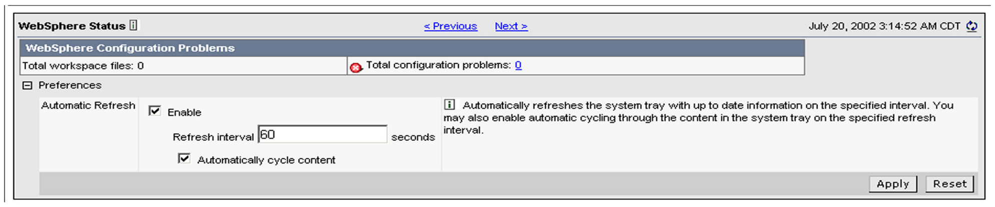


WebSphere Status < Previous Next > July 20, 2002 3:16:59 AM CDT

Websphere Runtime Messages			
Total All Messages:99	: 65 new, 65 total	: 0 new, 0 total	: 34 new, 34 total

⊞ Preferences

  - Configuration



WebSphere Status < Previous Next > July 20, 2002 3:14:52 AM CDT

WebSphere Configuration Problems	
Total workspace files: 0	Total configuration problems: 0

⊟ Preferences

Automatic Refresh ☑ Enable

Refresh interval 60 seconds

☑ Automatically cycle content

Automatically refreshes the system tray with up to date information on the specified interval. You may also enable automatic cycling through the content in the system tray on the specified refresh interval.

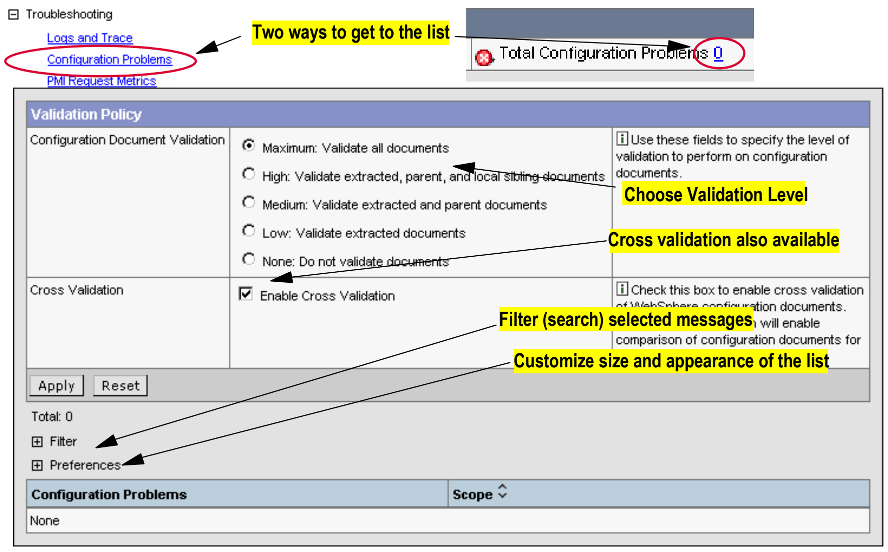
- **Manual or automatic refresh**
  - Configurable interval and content cycle

The console has been completely redesigned in Version 5, and also the way messages are reported to the console is new.

There is a clear separation between config messages and runtime messages - they are accessible in different lists.

You may enable automatic refresh - on by default - and configure the frequency of refresh operations.

Configuration Messages - List and Options


Two ways to get to the list
Choose Validation Level
Cross validation also available
Filter (search) selected messages
Customize size and appearance of the list

Troubleshooting
- Logs and Trace
- Configuration Problems
- PMI Request Metrics

Total Configuration Problems 0

Validation Policy

Configuration Document Validation	(●) Maximum: Validate all documents ○ High: Validate extracted, parent, and local sibling documents ○ Medium: Validate extracted and parent documents ○ Low: Validate extracted documents ○ None: Do not validate documents	Use these fields to specify the level of validation to perform on configuration documents.
Cross Validation	✓ Enable Cross Validation	Check this box to enable cross validation of WebSphere configuration documents. … will enable comparison of configuration documents for

Apply Reset

Total: 0

⊞ Filter

⊞ Preferences

Configuration Problems	Scope
None	

5 | Problem Determination Facilities | © 2002, 2003 IBM Corporation

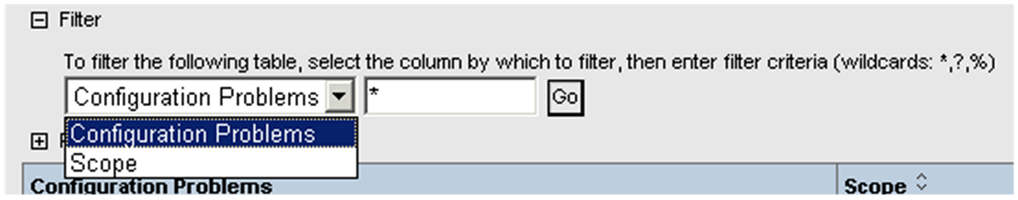
To access the config messages you can either click on the number of messages or click on the "Check for Configuration Problems" in the Problem Determination folder.

There are a number of levels of validation that you can choose from - the highest the level, the more cross-referencing is performed by the validators and the more information you will obtain.

The filters allow you to limit the number of messages displayed on the screen - as we explain in the next chart.

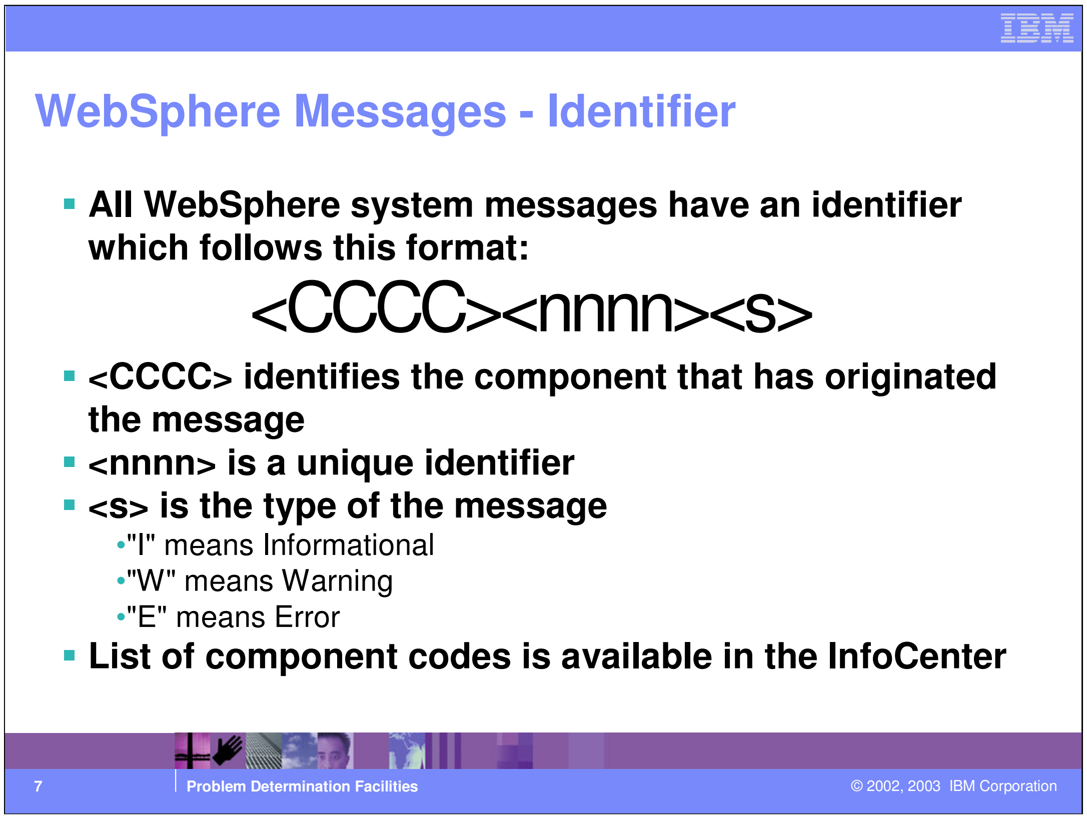
Filtering Configuration Problems

- **Filter based on message contents**
  - Header text will be searched for pattern
  - Wildcards allowed
- **Filter based on scope**
  - Narrow the list to a node, server, cluster, or group thereof


Filter

To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *,?,%)

You can filter messages based on their textual content by specifying a search string - and you can restrict the list to messages that are relevant to a particular node or application server.



WebSphere Messages - Identifier

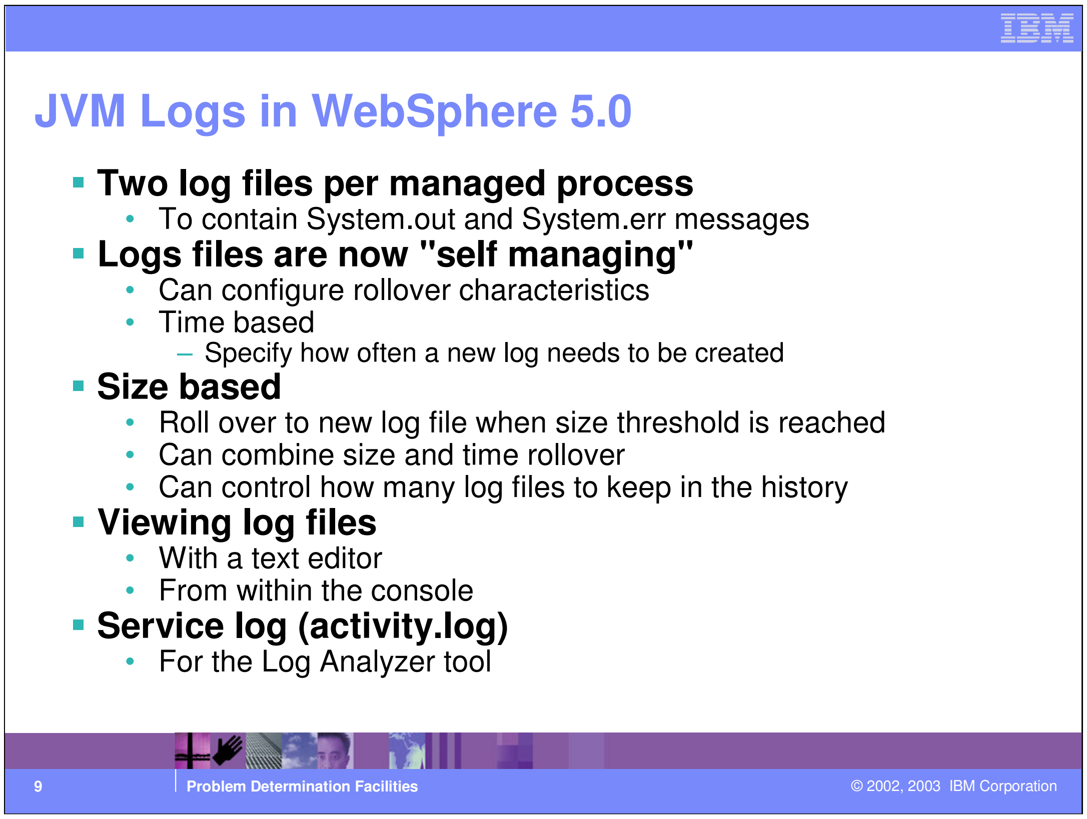
- **All WebSphere system messages have an identifier which follows this format:**

  <CCCC><nnnn><s>

- **<CCCC> identifies the component that has originated the message**
- **<nnnn> is a unique identifier**
- **<s> is the type of the message**
  - "I" means Informational
  - "W" means Warning
  - "E" means Error
- **List of component codes is available in the InfoCenter**

This is the structure of a WebSphere message - a complete list of components is going to be available in the InfoCenter.

Managing and Viewing Logs



JVM Logs in WebSphere 5.0

- **Two log files per managed process**
  - To contain System.out and System.err messages
- **Logs files are now "self managing"**
  - Can configure rollover characteristics
  - Time based
    - Specify how often a new log needs to be created
- **Size based**
  - Roll over to new log file when size threshold is reached
  - Can combine size and time rollover
  - Can control how many log files to keep in the history
- **Viewing log files**
  - With a text editor
  - From within the console
- **Service log (activity.log)**
  - For the Log Analyzer tool

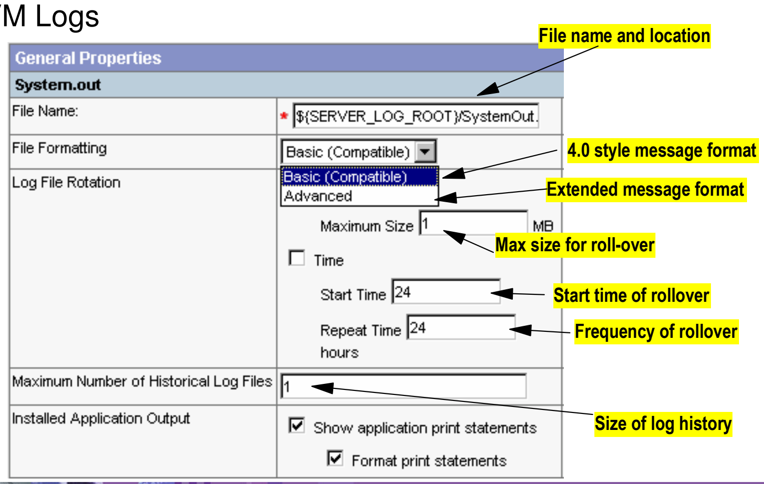
9 Problem Determination Facilities © 2002, 2003 IBM Corporation

New approach to managing logs in WebSphere 5.0: we have self-managed logs. You can configure the way these log files roll over and organize a log archiving procedure.

The criteria for rolling over are size or time base. You could combine the two, for example: you may want to have the log files roll over at 8 AM and at 8PM, unless they reached the size of 10MB in which case they would roll over no matter the time.

Configuring JVM Log Files

- **On the console, select a server and choose "Logging and Tracing"**
  - Click JVM Logs



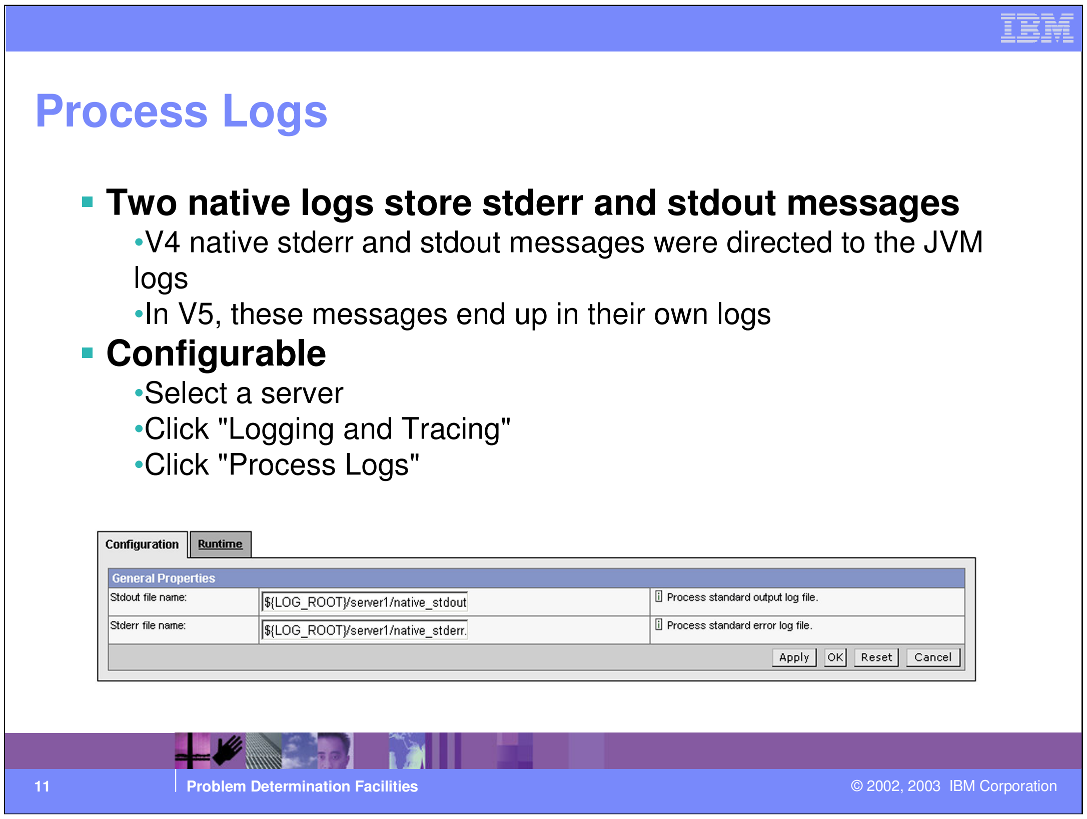
General Properties

System.out

File Name:	* ${SERVER_LOG_ROOT}/SystemOut.
File Formatting	Basic (Compatible) ▼ / Basic (Compatible) / Advanced
Log File Rotation	Maximum Size 1 MB; ☐ Time; Start Time 24; Repeat Time 24 hours
Maximum Number of Historical Log Files	1
Installed Application Output	☑ Show application print statements; ☑ Format print statements

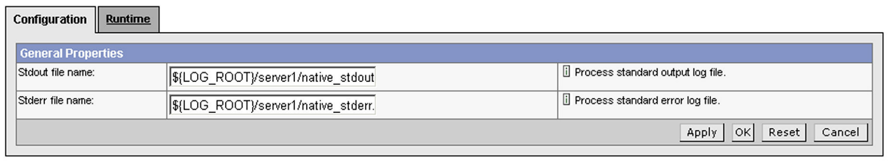
Logs are configurable from within the console as indicated in the chart.

You can get more information by selecting the Extended Message format - that won't be compatible with any utility you may have written to analyze 4.0 logs.



Process Logs

- **Two native logs store stderr and stdout messages**
  - •V4 native stderr and stdout messages were directed to the JVM logs
  - •In V5, these messages end up in their own logs
- **Configurable**
  - •Select a server
  - •Click "Logging and Tracing"
  - •Click "Process Logs"



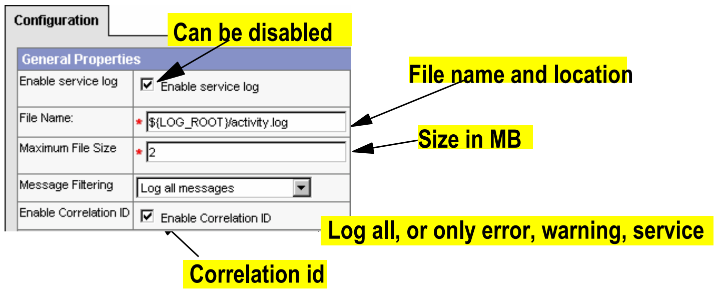
General Properties		
Stdout file name:	${LOG_ROOT}/server1/native_stdout	Process standard output log file.
Stderr file name:	${LOG_ROOT}/server1/native_stderr.	Process standard error log file.

11 Problem Determination Facilities © 2002, 2003 IBM Corporation

The process logs existed in 4.0. In 4.0, messages coming from both java code and native code were written to the process logs. In 5.0, we introduced the JVM logs and "separated" messages from java code and native code into two separate logs. We anticipate that under normal conditions, few, if any messages will get written to the process logs. Almost all messages will be written to the JVM logs. (We introduced the JVM logs so we could make the logs self-managing - much easier in java than in native code).

The IBM Service Log (activity.log)

- **Logs all messages directed to the individual log files**
  - System messages from all the app server
  - Additional service messages
  - Messages produced by instrumented applications (JRas)
- **Configurable**
  - Select server, click "Logging and Tracing", select "IBM Service Logs“
- **Viewable with the Log Analyzer**
  - ShowLog batch utility produces a formatted, readable version of the service log
  - Compressed format for performance

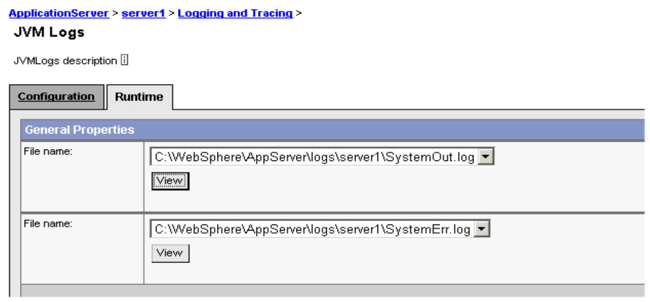


Configuration	
General Properties	
Enable service log	☑ Enable service log
File Name:	* ${LOG_ROOT}/activity.log
Maximum File Size	* 2
Message Filtering	Log all messages
Enable Correlation ID	☑ Enable Correlation ID

Essentially, the service (activity) log contains the same messages as are logged in the SystemOut JVM logs, plus a few extra serviceability messages. The key difference between the SystemOut JVM log and the service log is intended usage: We expect customers to view/archive the JVM logs to monitor the operation of the process, but when doing real problem determination, customers should use the service log. The service log is opened in the Log Analyzer. The Log Analyzer then uses a symptom database to analyze messages in the log and flags the ones related to known problems which have known solutions.

Viewing Log Files

- **JVM and Process Logs**
  - •Use text editor
    - –Notepad
  - •View from within the console
- **IBM service log**
  - •Use the Log Analyzer
  - •"Traditional" log analyzer tool shipped with WebSphere
  - •"Eclipse-based" log analyzer tool in WebSphere Studio App Developer


ApplicationServer > server1 > Logging and Tracing >

JVM Logs

JVMLogs description

File name: C:\WebSphere\AppServer\logs\server1\SystemOut.log

File name: C:\WebSphere\AppServer\logs\server1\SystemErr.log

Logs are also viewable from within the console.

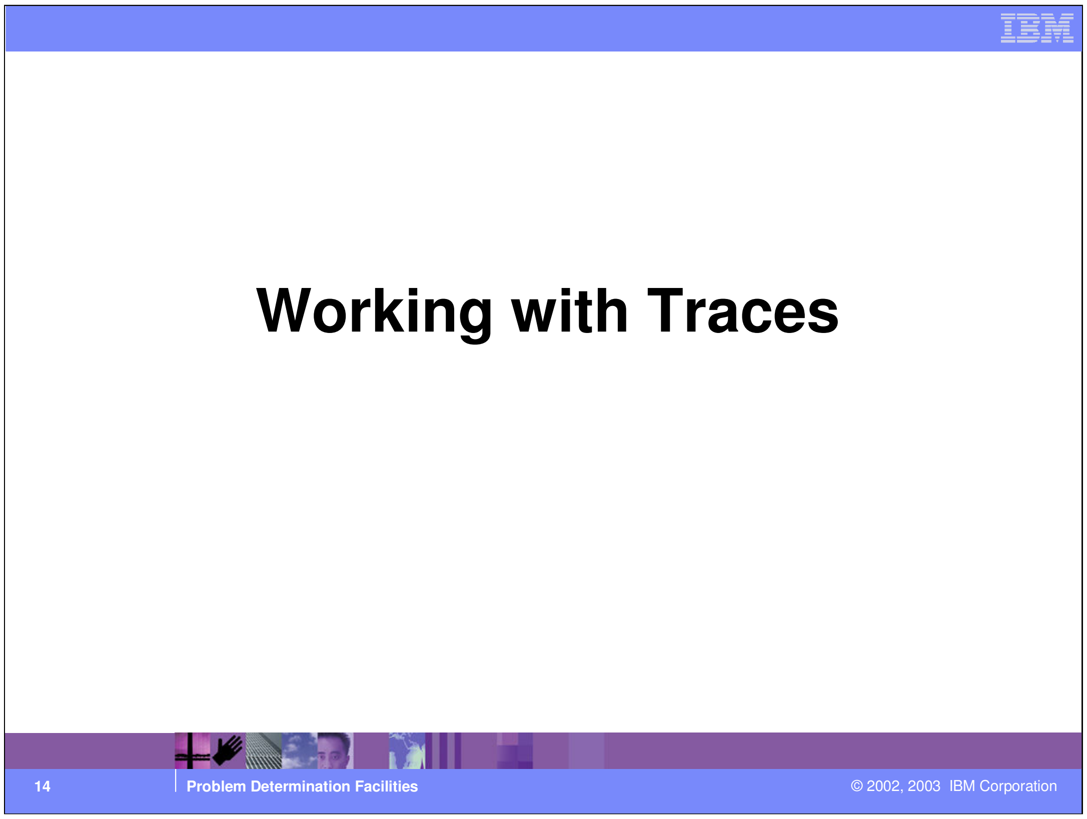
The service log is viewable with the log analyzer.

There will be two versions of the log analyze shipped with WebSphere 5.0

The traditional log analyzer as we know it from version 4

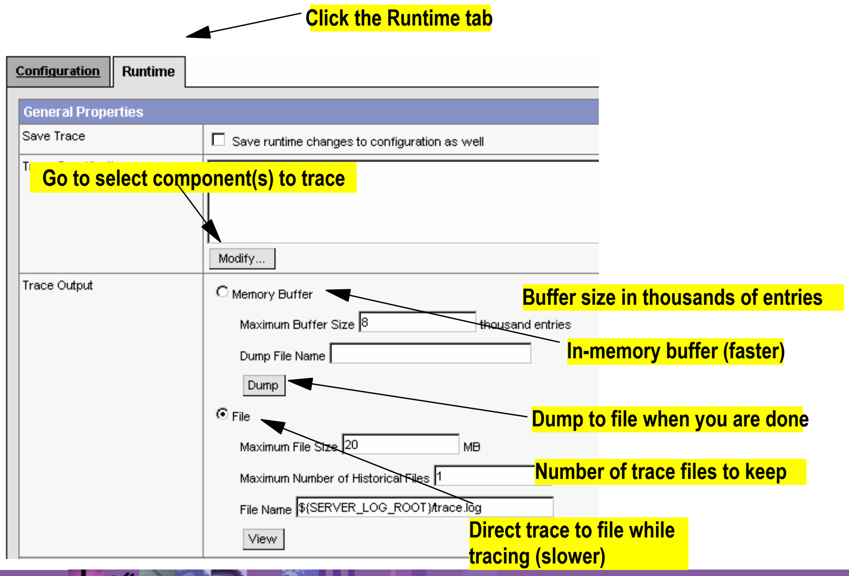
And a new log analyzer that is based on Eclipse and shipped as part of the Application Server ToolKit (ASTK)

Also - WebSphere Studio App Developer version 5 integrates the new Log Analyzer.



Working with Traces

Activating Traces on a Running Server


Click the Runtime tab
Go to select component(s) to trace
Buffer size in thousands of entries
In-memory buffer (faster)
Dump to file when you are done
Number of trace files to keep
Direct trace to file while tracing (slower)

Configuration | Runtime

General Properties

Save Trace: ☐ Save runtime changes to configuration as well

Modify...

Trace Output:
- ○ Memory Buffer
  - Maximum Buffer Size: 8 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

15 Problem Determination Facilities © 2002, 2003 IBM Corporation

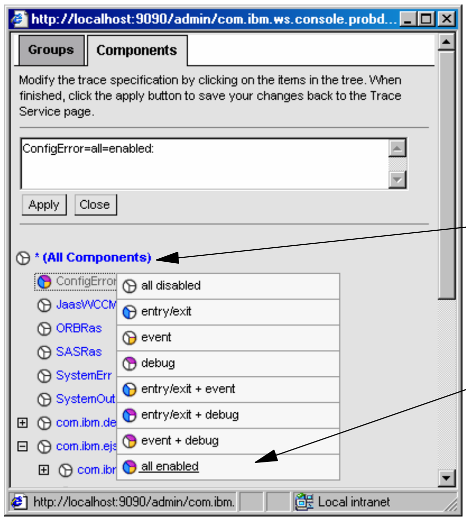
Traces can be activated by clicking on "Logging And Tracing" -> Diagnostic Trace on an application server's properties.

As in previous releases, you can activate a trace on a running server or you can choose to trace the startup of a server.

To trace a running server select the Runtime tab and then click on the Modify button to select the components you want to trace.

It's a good practice to store the trace in a circular buffer in memory (lesser impact on performance) and then dump it to a file to inspect it.

Selecting What Needs To Be Traced


http://localhost:9090/admin/com.ibm.ws.console.probd...

Groups | Components

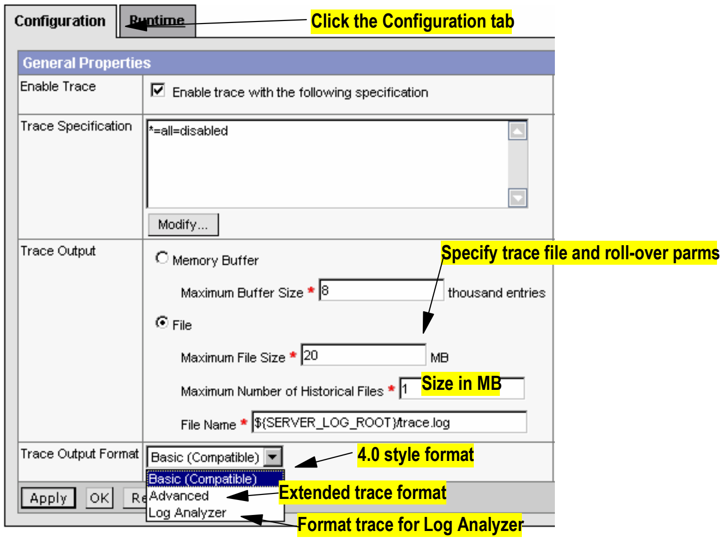
Modify the trace specification by clicking on the items in the tree. When finished, click the apply button to save your changes back to the Trace Service page.

```
ConfigError=all=enabled:
```

This is the new navigator window that allows you to select the components to trace and the type of trace.

The semantics haven't changed since version 4.

Tracing Servers from Startup



Configuration | Runtime

Click the Configuration tab

General Properties

Enable Trace: ☑ Enable trace with the following specification

Trace Specification: *=all=disabled

Modify...

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

Specify trace file and roll-over parms

Size in MB

Trace Output Format: Basic (Compatible)
Basic (Compatible)
Advanced
Log Analyzer

4.0 style format

Extended trace format

Format trace for Log Analyzer

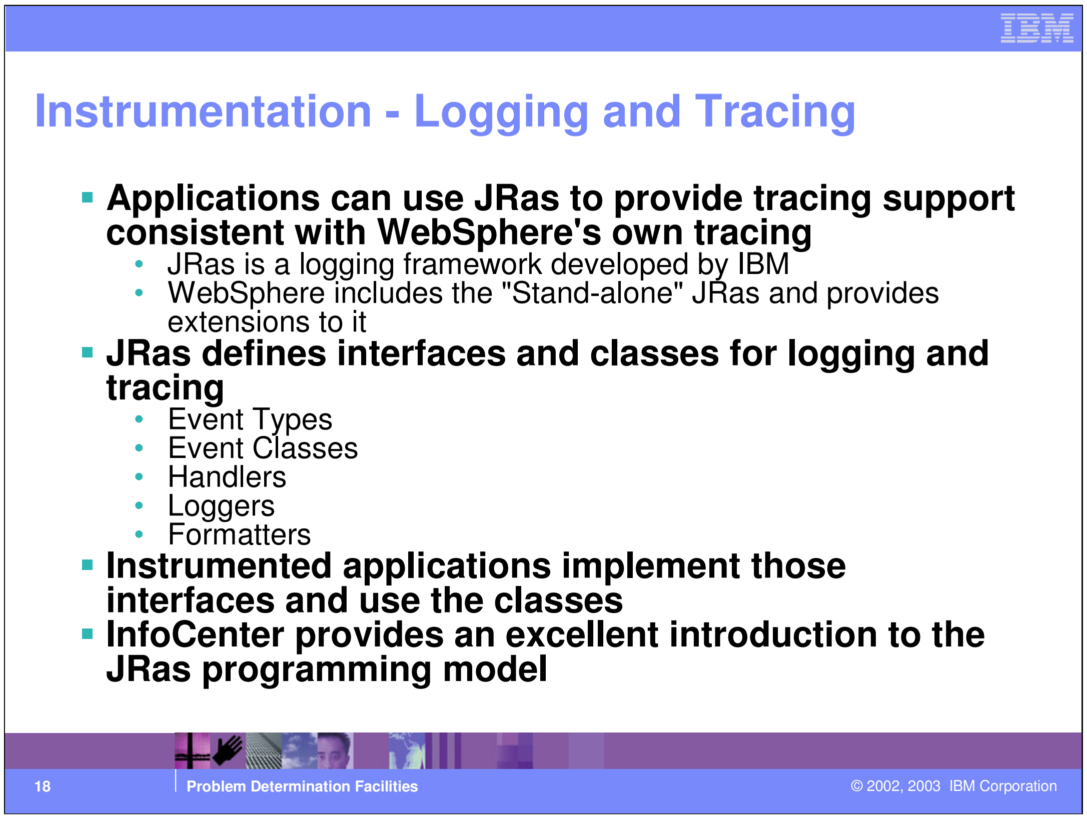
Apply | OK

Tracing a server from startup can be achieved by storing the trace settings in the configuration. Click on the Configuration tab and select that you want to enable the trace - then click on Modify to select the components to trace.

You can select to have the trace output sent directly to a file or stored into a memory buffer. The memory buffer is a circular buffer and can be safely dumped only if the application server completes the startup successfully (you would then open the Runtime tab and dump the buffer). If the application server crashes before the startup is complete, chances are that WebSphere won't be able to dump the buffer - and the trace would therefore be unavailable. Directing the startup trace to a file is slower, but it ensures that you'll get your trace output.

The rollover parameters allow you to determine how much trace information should be stored in an individual file and how many files need to be kept on line. For instance - if the you configured 10 MB for max file size and 5 for the number of files to be kept in the history - and the trace information takes 35 MB of disk space, you'll get 4 trace files (three of them will be 10MB in size, and the fourth and most recent would contain the latest 5 MB).

The format of the trace messages can be also configured to be either in line with the 4.0 style, or to provide additional information - a newer option, allows the trace to be formatted in a way that the log analyzer would be able to import.



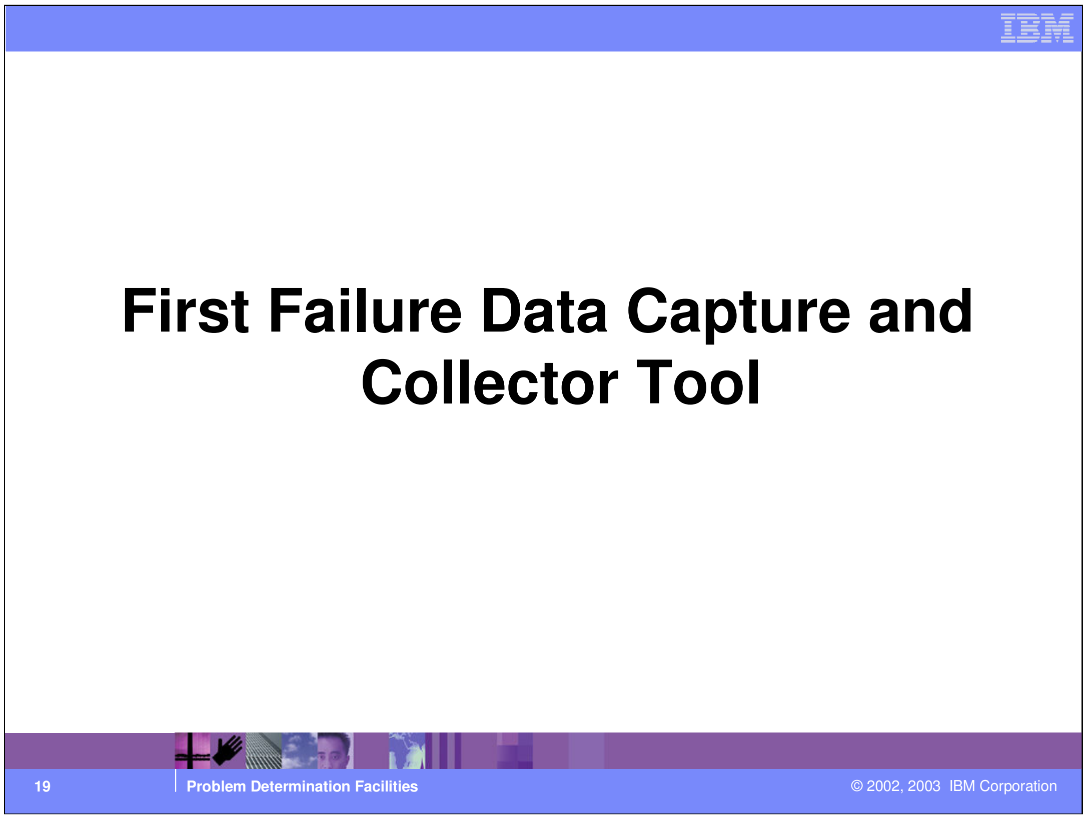
Instrumentation - Logging and Tracing

- **Applications can use JRas to provide tracing support consistent with WebSphere's own tracing**
  - JRas is a logging framework developed by IBM
  - WebSphere includes the "Stand-alone" JRas and provides extensions to it
- **JRas defines interfaces and classes for logging and tracing**
  - Event Types
  - Event Classes
  - Handlers
  - Loggers
  - Formatters
- **Instrumented applications implement those interfaces and use the classes**
- **InfoCenter provides an excellent introduction to the JRas programming model**

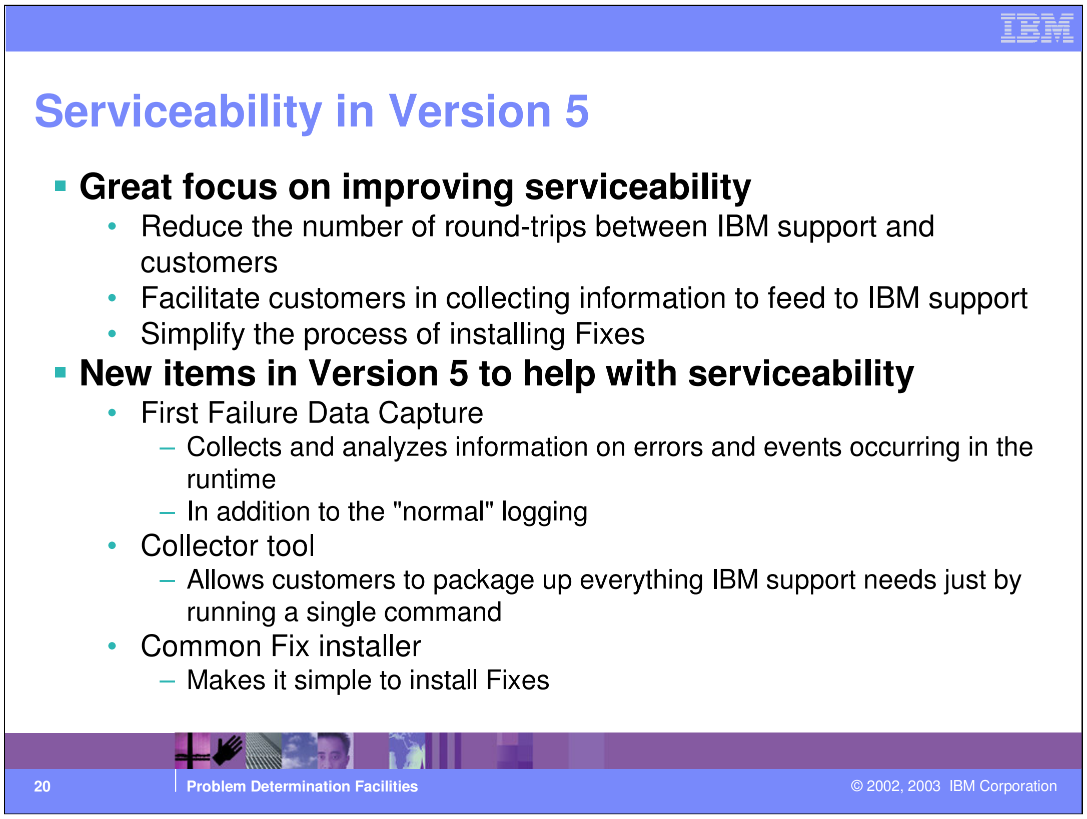
18 | Problem Determination Facilities © 2002, 2003 IBM Corporation

WebSphere allows customers and software providers to instrument their applications so that they can be traced and logged the same way as the WebSphere runtime

This is made possible through the use of the JRas framework - which was developed by IBM and it is include with some extensions in WebSphere.



First Failure Data Capture and Collector Tool



Serviceability in Version 5

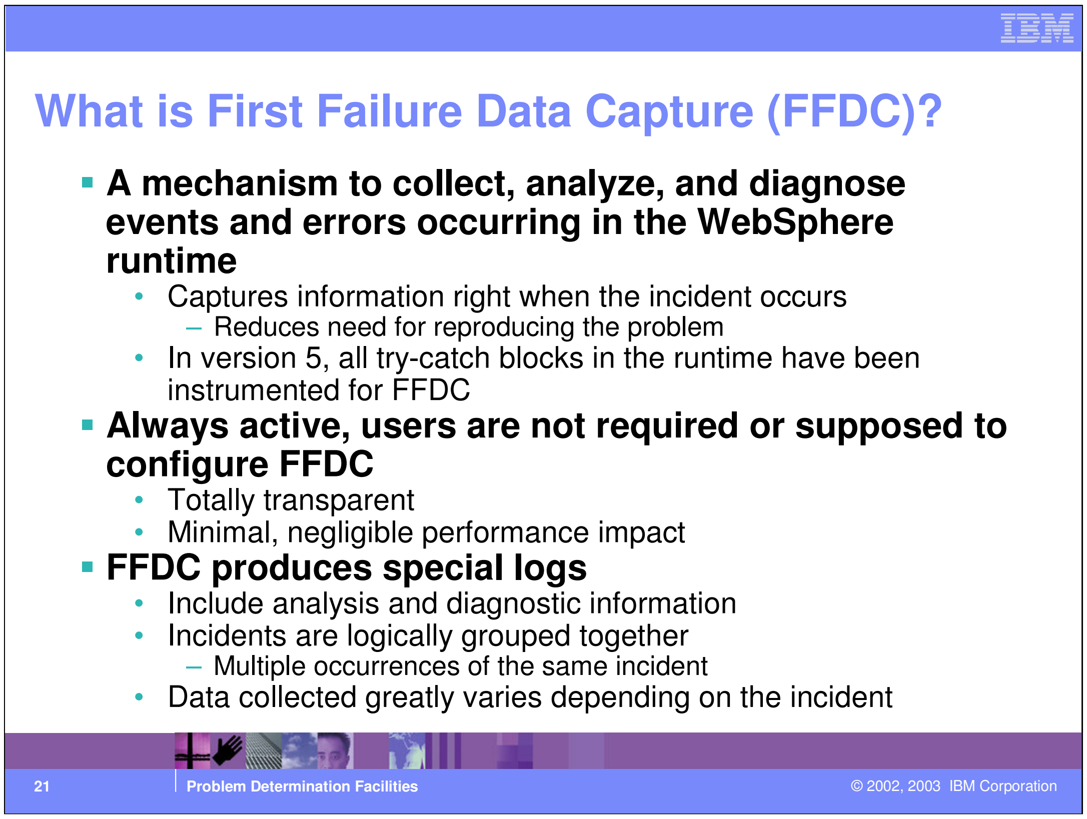
- **Great focus on improving serviceability**
  - Reduce the number of round-trips between IBM support and customers
  - Facilitate customers in collecting information to feed to IBM support
  - Simplify the process of installing Fixes
- **New items in Version 5 to help with serviceability**
  - First Failure Data Capture
    - Collects and analyzes information on errors and events occurring in the runtime
    - In addition to the "normal" logging
  - Collector tool
    - Allows customers to package up everything IBM support needs just by running a single command
  - Common Fix installer
    - Makes it simple to install Fixes

20 Problem Determination Facilities © 2002, 2003 IBM Corporation

Great effort went into make serviceability a priority item of Version 5.

The spirit of these efforts is to preemptively collect and diagnose problems, right when the problems occur, rather than taking the customer through a painful and time consuming process of problem recreation.

First failure data capture and the collector tool will greatly reduce the need for round trips between IBM Support and the customers.



What is First Failure Data Capture (FFDC)?

- **A mechanism to collect, analyze, and diagnose events and errors occurring in the WebSphere runtime**
  - Captures information right when the incident occurs
    - Reduces need for reproducing the problem
  - In version 5, all try-catch blocks in the runtime have been instrumented for FFDC
- **Always active, users are not required or supposed to configure FFDC**
  - Totally transparent
  - Minimal, negligible performance impact
- **FFDC produces special logs**
  - Include analysis and diagnostic information
  - Incidents are logically grouped together
    - Multiple occurrences of the same incident
  - Data collected greatly varies depending on the incident

21 Problem Determination Facilities © 2002, 2003 IBM Corporation

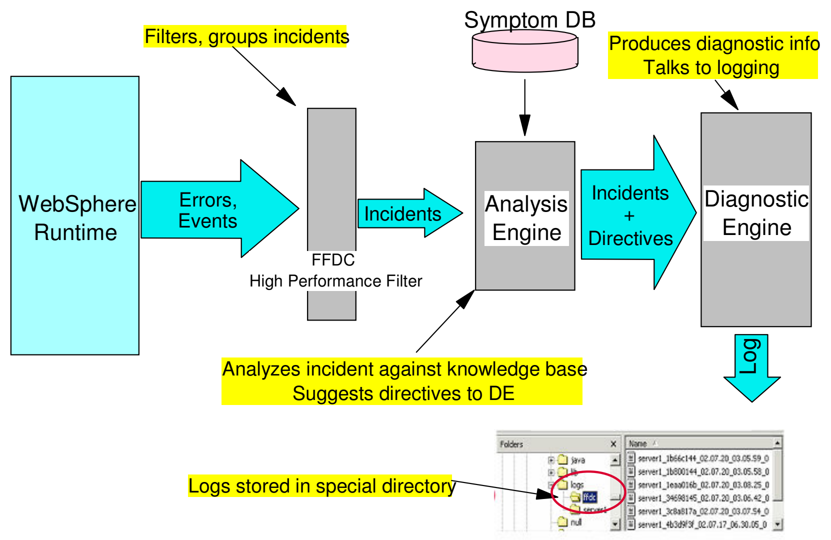
First failure data capture is a mechanism that collects and diagnoses exceptions, error conditions, failure, and potential problem situations - at the time they actually occur.

All the WebSphere runtime code is now instrumented to interact with FFDC - in particular all the try/catch blocks include FFDC-related code.

From a customer's standpoint, there is nothing to configure, turn on, or modify to be covered by FFDC support.

FFDC will produce special log files that are meant to be looked at by IBM qualified personnel.

How does FFDC works?

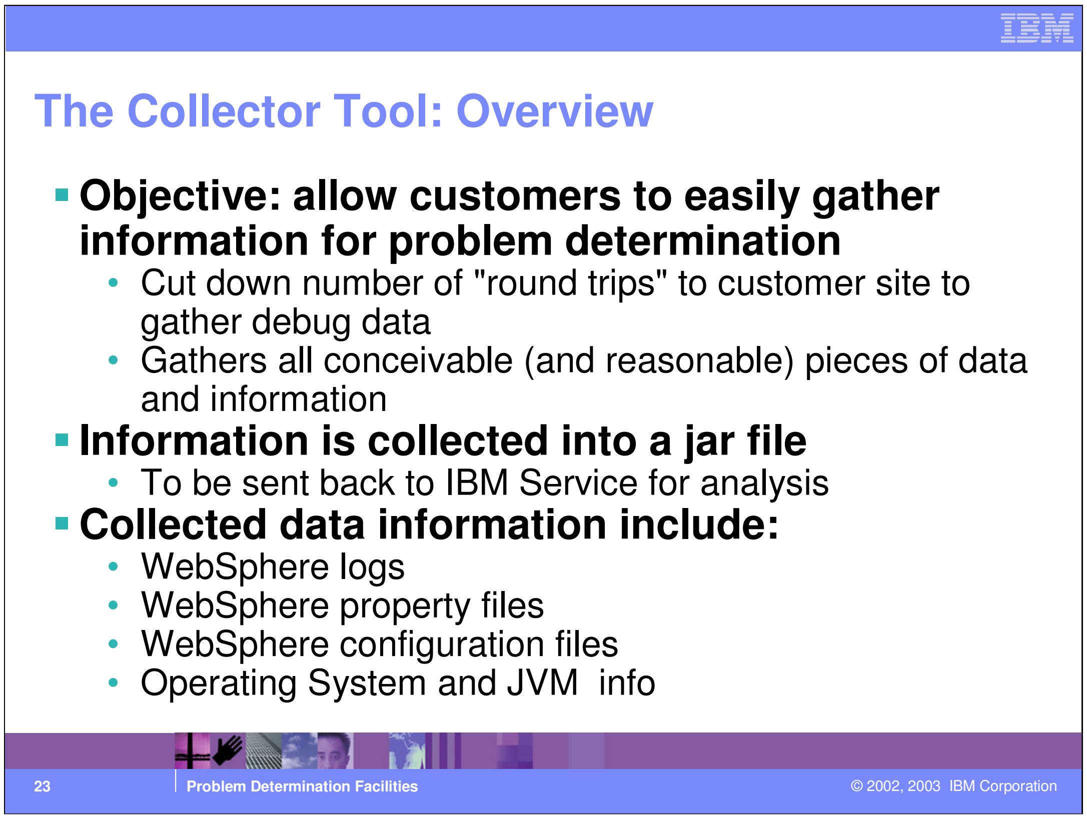

Filters, groups incidents
Symptom DB
Produces diagnostic info
Talks to logging
WebSphere Runtime
Errors, Events
Incidents
FFDC
High Performance Filter
Analysis Engine
Incidents + Directives
Diagnostic Engine
Log
Analyzes incident against knowledge base
Suggests directives to DE
Logs stored in special directory
Folders
java
lib
logs
ffdc
server1
null
Name
server1_1b66c144_02.07.20_03.05.59_0
server1_1b800144_02.07.20_03.05.58_0
server1_1eaa016b_02.07.20_03.08.25_0
server1_34698145_02.07.20_03.06.42_0
server1_3c8a817a_02.07.20_03.07.54_0
server1_4b3d9f3f_02.07.17_06.30.05_0

This is the architecture of the FFDC engine. The instrumented runtime produces errors and events - which are then filtered by the "high performance filter" (HPF)

The HPF creates incidents based on the event data that is produced by the runtime. The HPF is smart enough to recognize whether a certain event is a duplicate and increment a counter of occurrences in that case, rather than creating a new incident.

The Analysis Engine (AE) has the primary function of directing the activities of the diagnostic engine. The AE uses the Log Analyzer symptom database to influence the diagnostic engine, by suggesting which information needs to be collected and logged and from which components, based on the type of incident. For example, a database related incident may require that the Diagnostic Engine (DE) collects and logs information about the SQLCODE from the JDBC layer - a security incident may require a completely different set of data and may involve different components.

The logs produced by the DE are stored in a special FFDC directory.

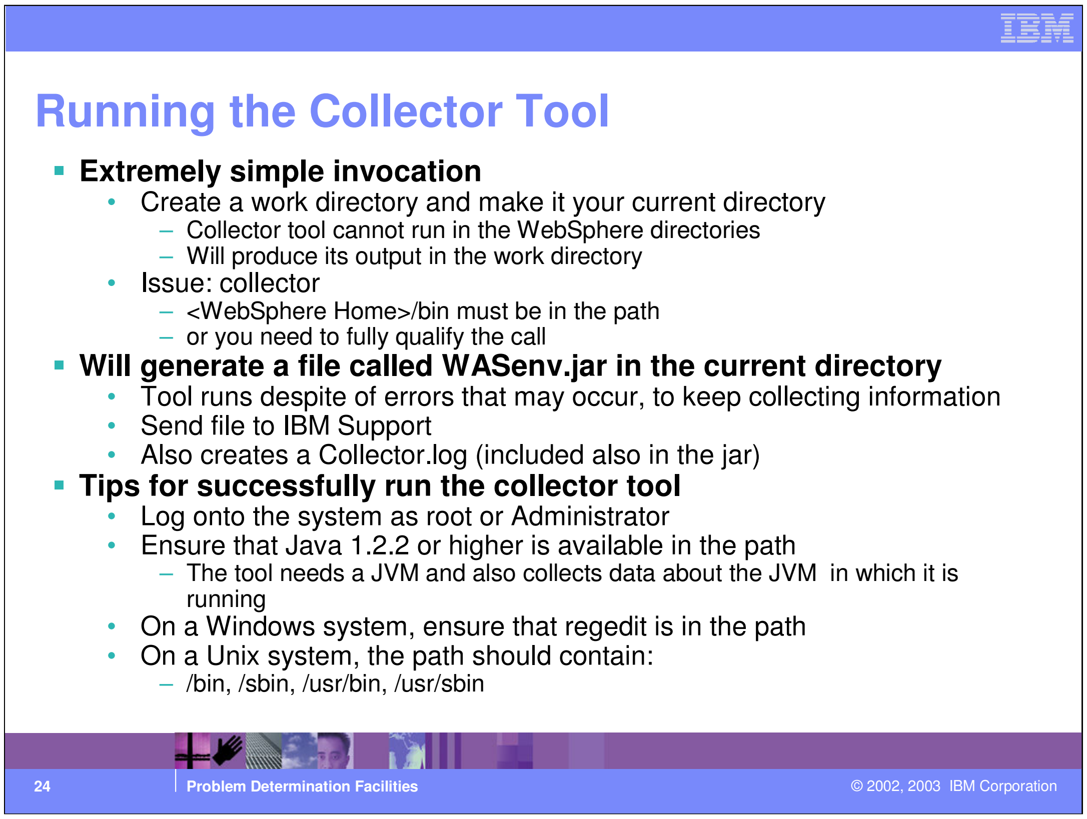


The Collector Tool: Overview

- **Objective: allow customers to easily gather information for problem determination**
  - Cut down number of "round trips" to customer site to gather debug data
  - Gathers all conceivable (and reasonable) pieces of data and information
- **Information is collected into a jar file**
  - To be sent back to IBM Service for analysis
- **Collected data information include:**
  - WebSphere logs
  - WebSphere property files
  - WebSphere configuration files
  - Operating System and JVM info

The collector tool allows a customer to run a single command that gathers all the necessary information that IBM Support may need to analyze a problem.

The customer is only supposed to run the command - the tool will produce a jar file that IBM Support will then inspect.



Running the Collector Tool

- **Extremely simple invocation**
  - Create a work directory and make it your current directory
    - Collector tool cannot run in the WebSphere directories
    - Will produce its output in the work directory
  - Issue: collector
    - <WebSphere Home>/bin must be in the path
    - or you need to fully qualify the call
- **Will generate a file called WASenv.jar in the current directory**
  - Tool runs despite of errors that may occur, to keep collecting information
  - Send file to IBM Support
  - Also creates a Collector.log (included also in the jar)
- **Tips for successfully run the collector tool**
  - Log onto the system as root or Administrator
  - Ensure that Java 1.2.2 or higher is available in the path
    - The tool needs a JVM and also collects data about the JVM in which it is running
  - On a Windows system, ensure that regedit is in the path
  - On a Unix system, the path should contain:
    - /bin, /sbin, /usr/bin, /usr/sbin

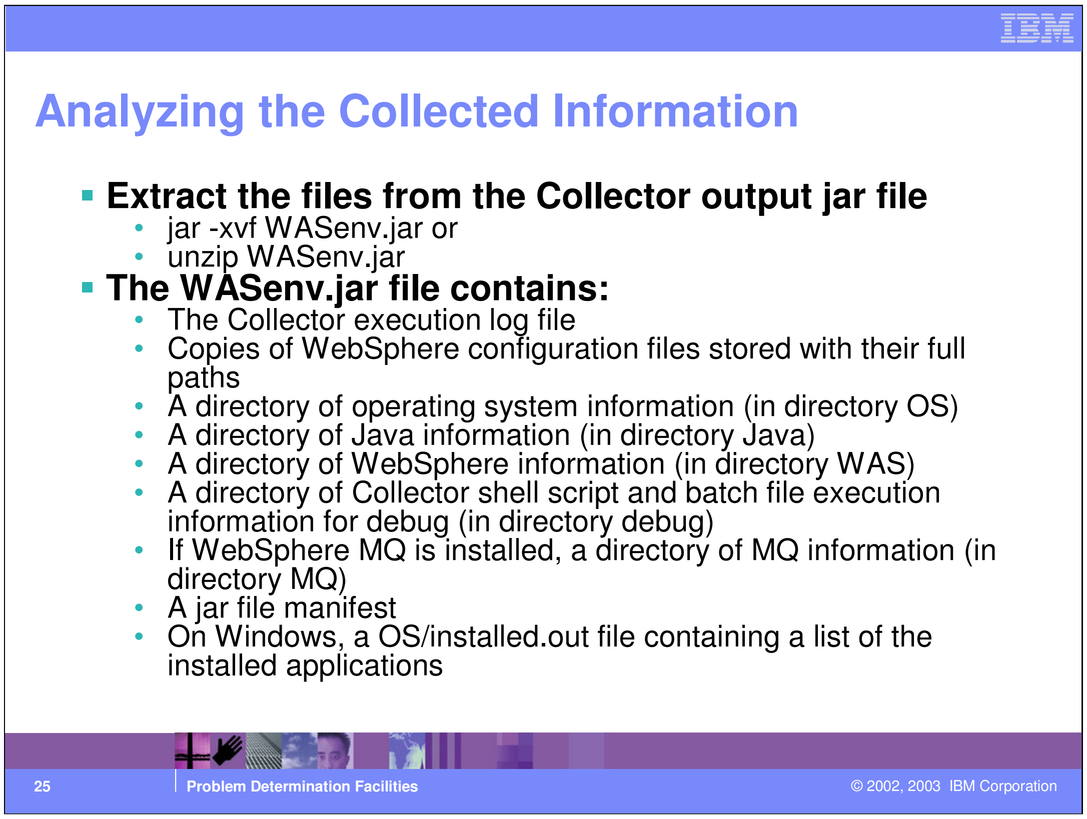
24 Problem Determination Facilities © 2002, 2003 IBM Corporation

The syntax of launching the collector tool is very simple. The tool will run and try to collect as much information as possible, even if it incurs into errors or if it doesn't find all the information it was expecting to gather.

However - there are some tips that you need to keep in mind - such as logging on as Administrator or root, making sure that Java is accessible, and so on.

Most importantly- you need to create a work directory and make it your current directory. If you attempt to run the collector tool when your current directory is WebSphere\appserver\bin, for instance, the tool won't run. This is to avoid file collisions and problems, since the collector tool produces a file with a fixed name (WASenv.jar) in the current directory.

In order to successfully run the collector tool, <WAS HOME>/bin needs to be in the path (or you can fully qualify the call to the batch file - or shell file). Also - the collector tool needs a Java Virtual Machine in order to run - we support the IBM Developer Kit, Java Technology Edition, Version 1.2.2 and later levels. However - since the collector tool tries to collect information on the Developer Kit as well, it's important that you make sure that you are using WebSphere's Developer Kit to run the collector tool (<WAS HOME>\java\bin) if you want to include information about the Developer Kit being used by WebSphere.

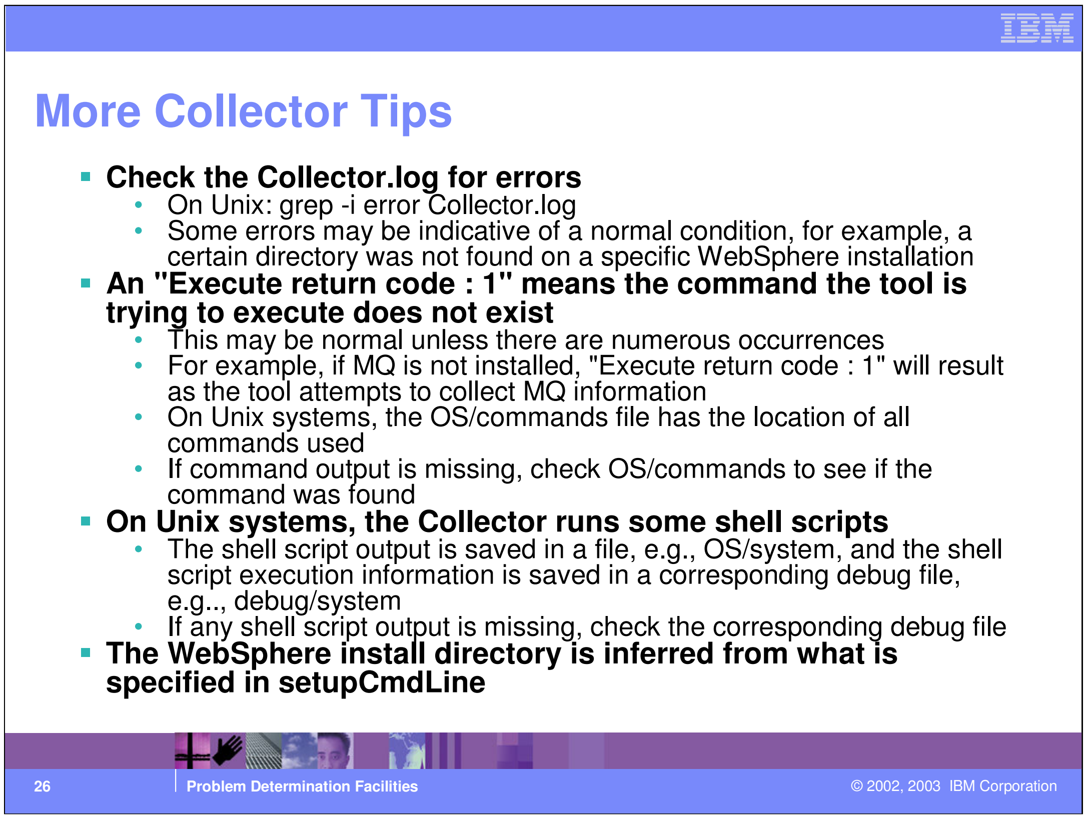


Analyzing the Collected Information

- **Extract the files from the Collector output jar file**
  - jar -xvf WASenv.jar or
  - unzip WASenv.jar
- **The WASenv.jar file contains:**
  - The Collector execution log file
  - Copies of WebSphere configuration files stored with their full paths
  - A directory of operating system information (in directory OS)
  - A directory of Java information (in directory Java)
  - A directory of WebSphere information (in directory WAS)
  - A directory of Collector shell script and batch file execution information for debug (in directory debug)
  - If WebSphere MQ is installed, a directory of MQ information (in directory MQ)
  - A jar file manifest
  - On Windows, a OS/installed.out file containing a list of the installed applications

25 Problem Determination Facilities © 2002, 2003 IBM Corporation

The task of analyzing the output of the collector tool is up to the IBM Support personnel. This chart outlines some recommendations on how to get started with the analysis.



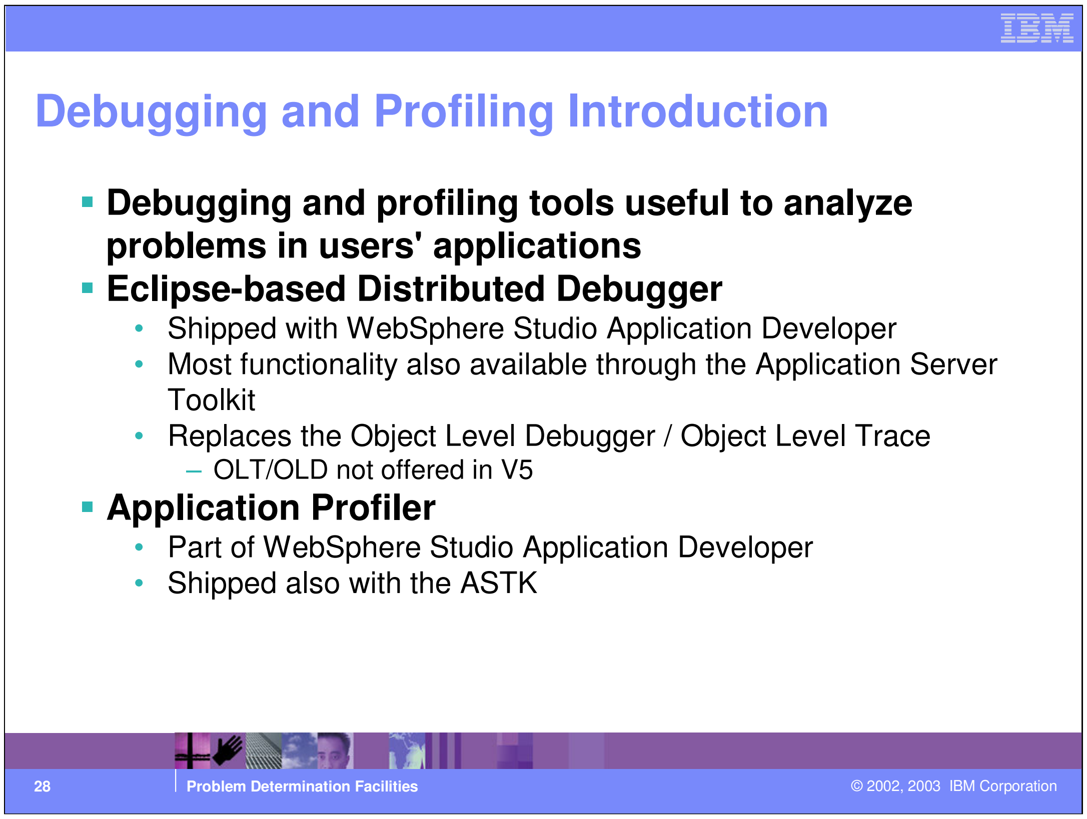
More Collector Tips

- **Check the Collector.log for errors**
  - On Unix: grep -i error Collector.log
  - Some errors may be indicative of a normal condition, for example, a certain directory was not found on a specific WebSphere installation
- **An "Execute return code : 1" means the command the tool is trying to execute does not exist**
  - This may be normal unless there are numerous occurrences
  - For example, if MQ is not installed, "Execute return code : 1" will result as the tool attempts to collect MQ information
  - On Unix systems, the OS/commands file has the location of all commands used
  - If command output is missing, check OS/commands to see if the command was found
- **On Unix systems, the Collector runs some shell scripts**
  - The shell script output is saved in a file, e.g., OS/system, and the shell script execution information is saved in a corresponding debug file, e.g.., debug/system
  - If any shell script output is missing, check the corresponding debug file
- **The WebSphere install directory is inferred from what is specified in setupCmdLine**

26 Problem Determination Facilities © 2002, 2003 IBM Corporation

The collector tool produces its own log - it may be interesting to look at the log to verify that the tool has actually gathered all the information it was expecting to find. In some cases, however, error messages logged in the collector log may not be the indication that something is wrong - they may just be a normal consequence of the particular environment where the tool was run.

Debugging and Profiling



Debugging and Profiling Introduction

- **Debugging and profiling tools useful to analyze problems in users' applications**
- **Eclipse-based Distributed Debugger**
  - Shipped with WebSphere Studio Application Developer
  - Most functionality also available through the Application Server Toolkit
  - Replaces the Object Level Debugger / Object Level Trace
    - OLT/OLD not offered in V5
- **Application Profiler**
  - Part of WebSphere Studio Application Developer
  - Shipped also with the ASTK

28 Problem Determination Facilities © 2002, 2003 IBM Corporation

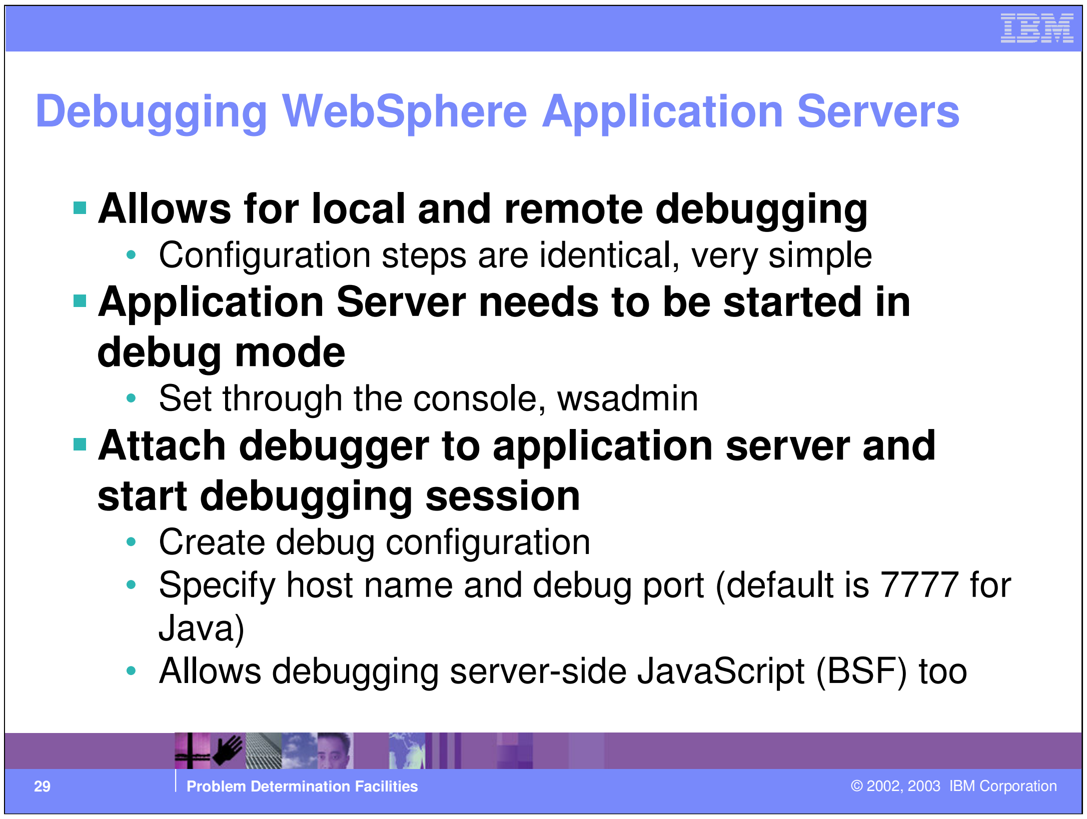
Traces, logs, FFDC, etc. are powerful tools to troubleshoot primarily WebSphere configuration and runtime problems.

However - in some cases, the problem resides in the applications code.

Debuggers and profilers allow to inspect, step through, and analyze the behavior of user written code.

In Version 5, WebSphere offers a distributed debugger and an application profiler - they are eclipse-based tools packaged in the AST (Application Server Toolkit).

The same tools, enhanced with additional features, are also available through the WebSphere Studio Application Developer product.



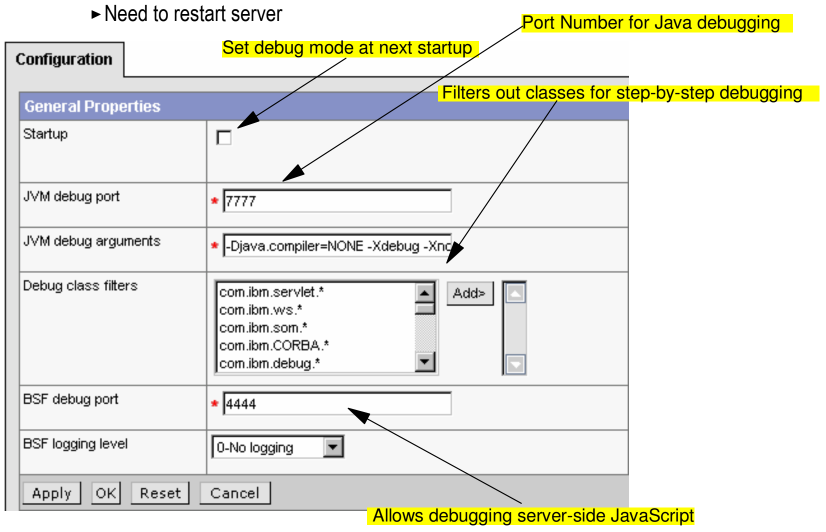
Debugging WebSphere Application Servers

- **Allows for local and remote debugging**
  - Configuration steps are identical, very simple
- **Application Server needs to be started in debug mode**
  - Set through the console, wsadmin
- **Attach debugger to application server and start debugging session**
  - Create debug configuration
  - Specify host name and debug port (default is 7777 for Java)
  - Allows debugging server-side JavaScript (BSF) too

29 Problem Determination Facilities © 2002, 2003 IBM Corporation

Configuring Application Server for Debugging

- Click on <server>->Debugging Service
  - Need to restart server



Set debug mode at next startup

Port Number for Java debugging

Filters out classes for step-by-step debugging

Allows debugging server-side JavaScript

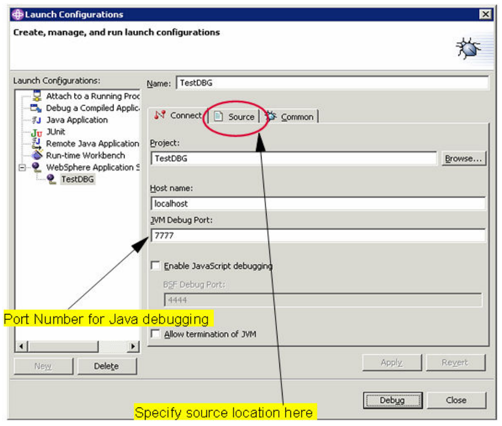
Configuration

General Properties

Startup	☐
JVM debug port	* 7777
JVM debug arguments	* -Djava.compiler=NONE -Xdebug -Xnc
Debug class filters	com.ibm.servlet.* com.ibm.ws.* com.ibm.som.* com.ibm.CORBA.* com.ibm.debug.*
BSF debug port	* 4444
BSF logging level	0-No logging

Configuring Your Debugging Session

- **Start WebSphere Studio or the Application Server Toolkit**
- **Create a Java project and import the source code you want to debug**
  - You can also point to external locations or wait to be prompted for source code location
- **Launch a debug configuration, selecting "WebSphere Application Server Debug"**



Launch Configurations

Create, manage, and run launch configurations

Launch Configurations:
- Attach to a Running Proc
- Debug a Compiled Applic
- Java Application
- JUnit
- Remote Java Application
- Run-time Workbench
- WebSphere Application S
  - TestDBG

Name: TestDBG

Project: TestDBG

Host name: localhost

JVM Debug Port: 7777

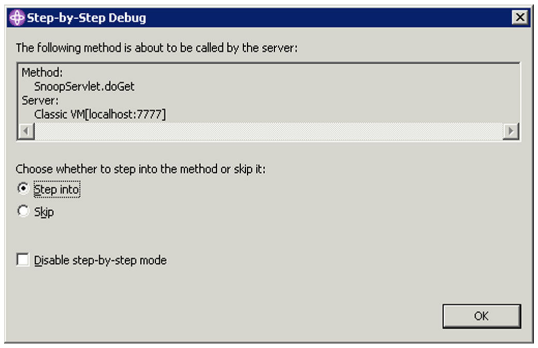
☐ Enable JavaScript debugging

BSF Debug Port: 4444

☐ Allow termination of JVM

Step-by-step Debugging

- **Stops at the boundaries of Web Components**
- **Allows you to step into the code or to skip**
- **Can be disabled**
  - Very useful to initiate your debugging session if you don't have breakpoints set



The following method is about to be called by the server:

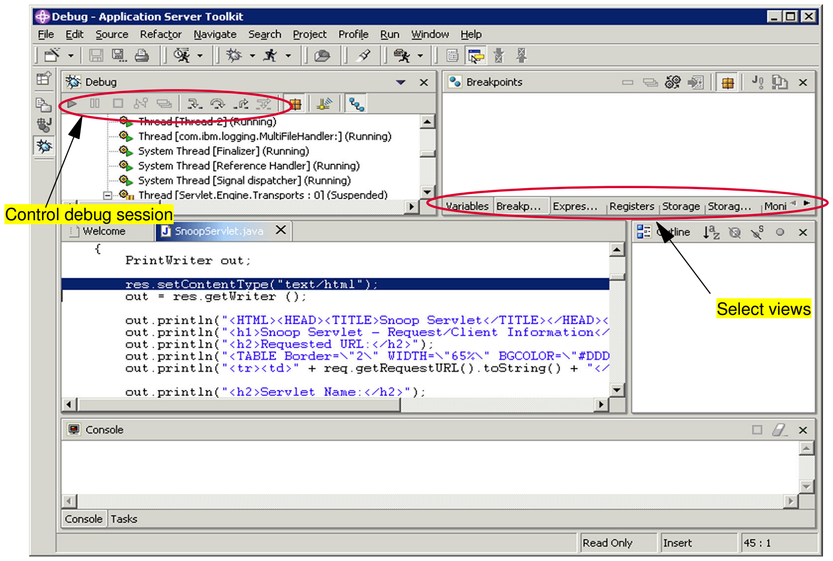
```
Method:
   SnoopServlet.doGet
Server:
   Classic VM[localhost:7777]
```

Choose whether to step into the method or skip it:

[✓ Step into]
Skip

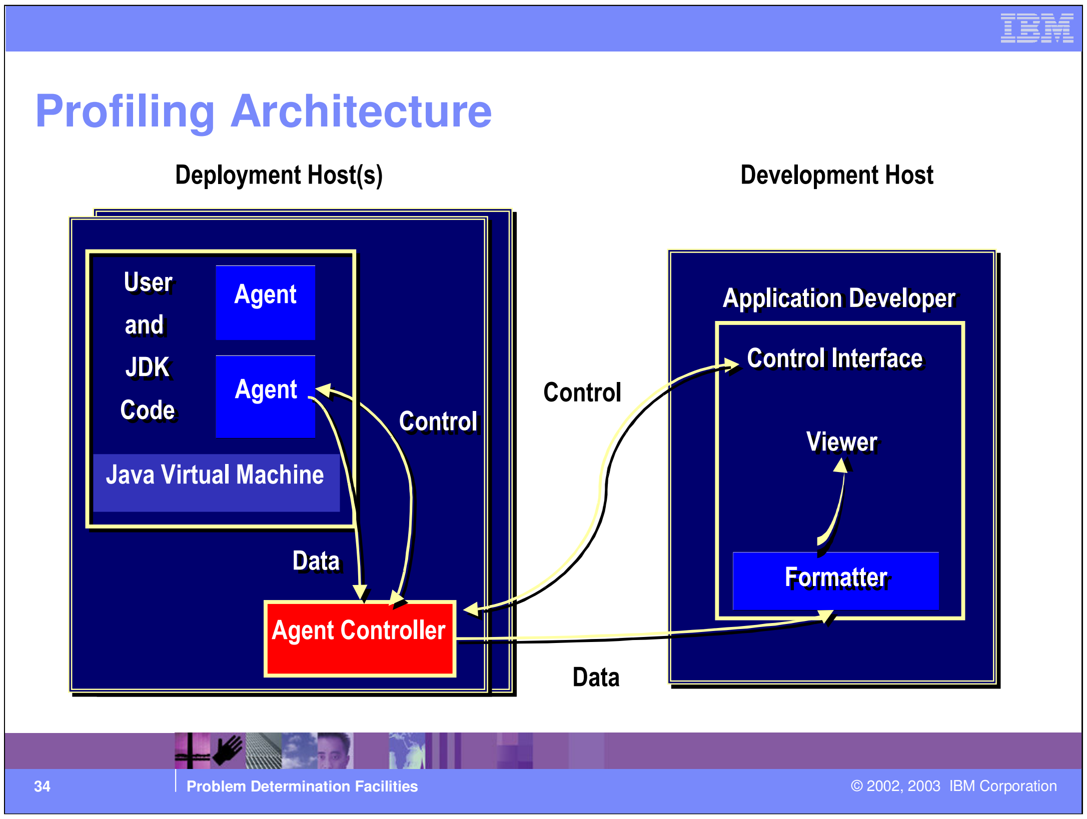
☐ Disable step-by-step mode

Debugging Source Code

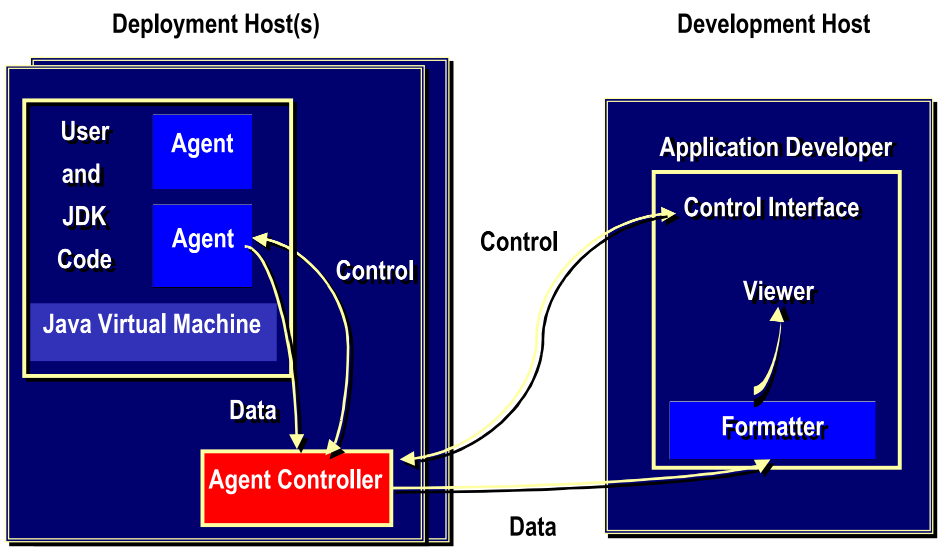

Debug - Application Server Toolkit
File Edit Source Refactor Navigate Search Project Profile Run Window Help
Debug
Breakpoints
Variables Breakp... Expres... Registers Storage Storag... Moni
Welcome
SnoopServlet.java
Outline
Console
Console Tasks
Read Only
Insert
45 : 1
Control debug session
Select views

```
Thread [Thread-2] (Running)
Thread [com.ibm.logging.MultiFileHandler:] (Running)
System Thread [Finalizer] (Running)
System Thread [Reference Handler] (Running)
System Thread [Signal dispatcher] (Running)
Thread [Servlet.Engine.Transports : 0] (Suspended)
```

```
    {
        PrintWriter out;

        res.setContentType("text/html");
        out = res.getWriter ();

        out.println("<HTML><HEAD><TITLE>Snoop Servlet</TITLE></HEAD><
        out.println("<h1>Snoop Servlet - Request/Client Information</
        out.println("<h2>Requested URL:</h2>");
        out.println("<TABLE Border=\"2\" WIDTH=\"65%\" BGCOLOR=\"#DDD
        out.println("<tr><td>" + req.getRequestURL().toString() + "</

        out.println("<h2>Servlet Name:</h2>");
```

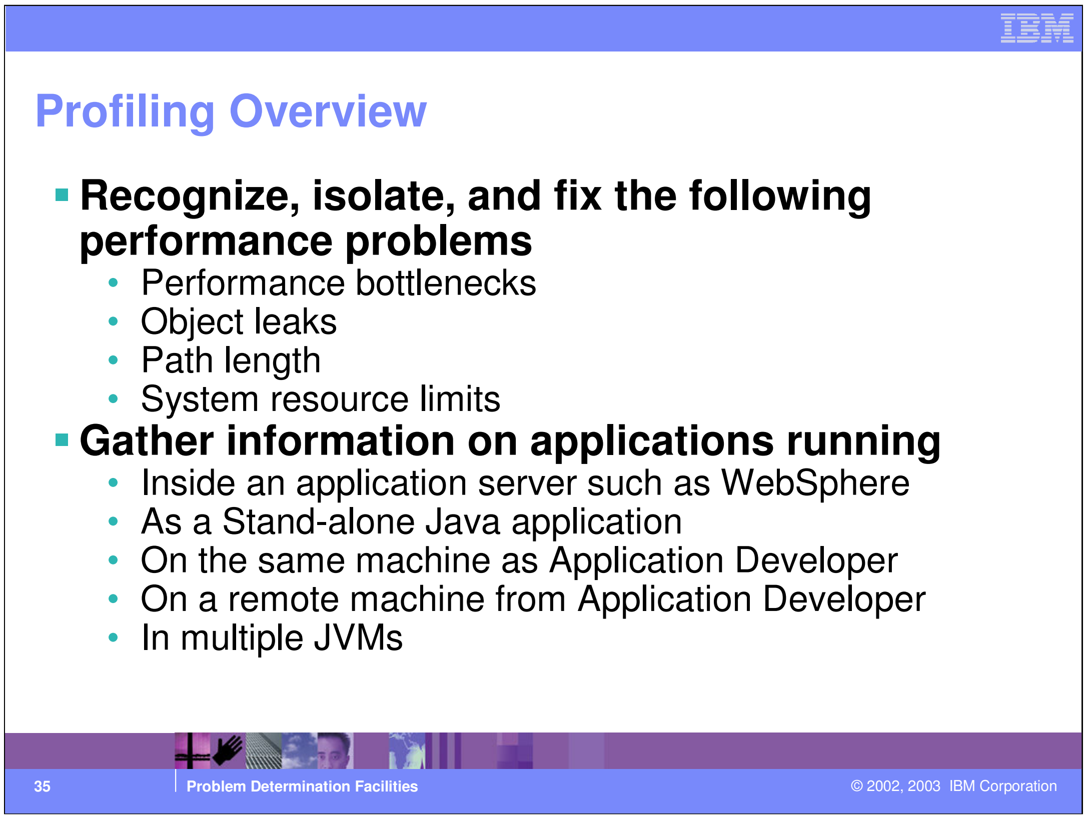


Profiling Architecture



The main ingredient to the architecture is the remote agent controller known as the IBM Agent Controller.

The IBM Agent Controller is installed on the deployment host and interacts with the JVM agents, such as the profiling agents.. Then the IBM Agent Controller takes the data and sends this to Application Developer (or eclipse-based profiler/debugger) where the information can be viewed and analyzed.

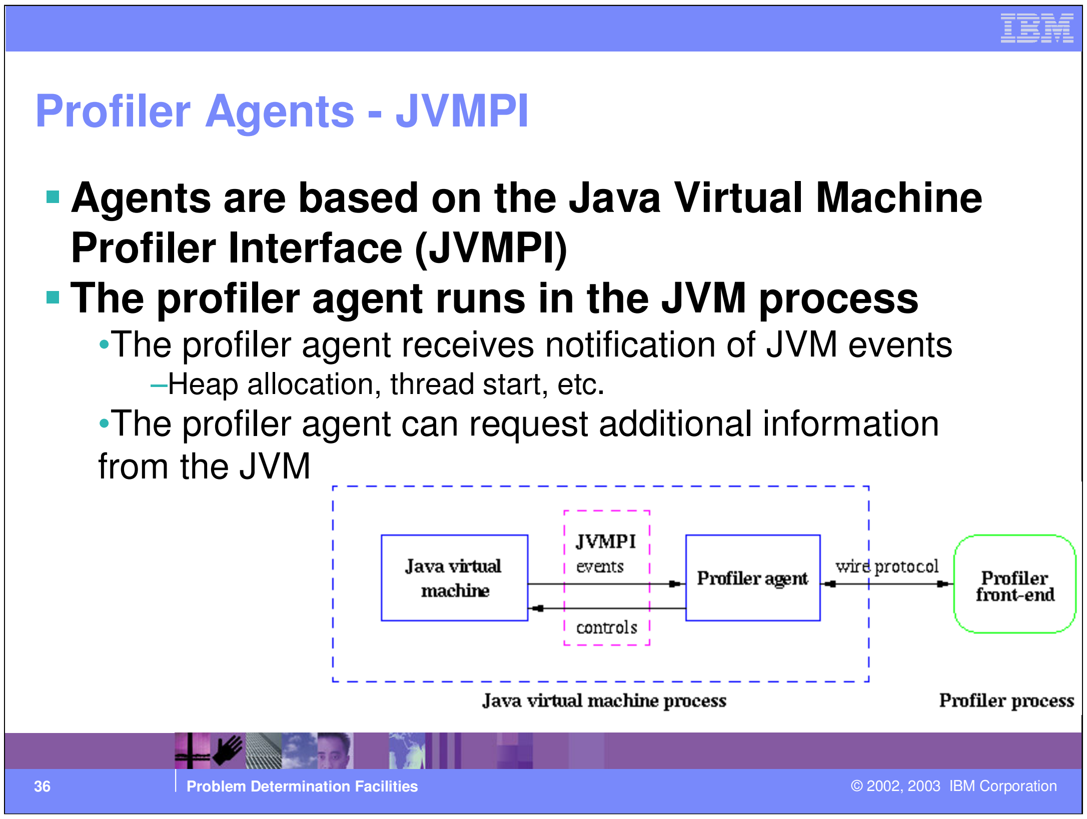


Profiling Overview

- **Recognize, isolate, and fix the following performance problems**
  - Performance bottlenecks
  - Object leaks
  - Path length
  - System resource limits
- **Gather information on applications running**
  - Inside an application server such as WebSphere
  - As a Stand-alone Java application
  - On the same machine as Application Developer
  - On a remote machine from Application Developer
  - In multiple JVMs

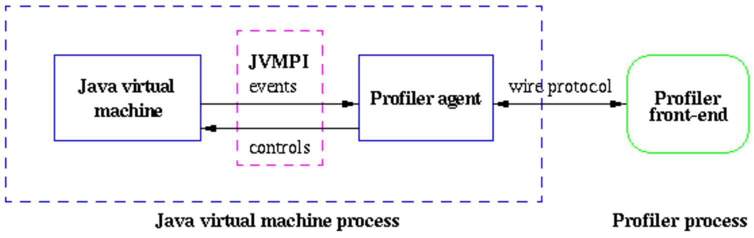
35 Problem Determination Facilities © 2002, 2003 IBM Corporation

The WSAD allows developers to start performance testing in the development stage so as not to have as many critical situations right before production.



Profiler Agents - JVMPI

- **Agents are based on the Java Virtual Machine Profiler Interface (JVMPI)**
- **The profiler agent runs in the JVM process**
  - •The profiler agent receives notification of JVM events
    - –Heap allocation, thread start, etc.
  - •The profiler agent can request additional information from the JVM



36 | Problem Determination Facilities | © 2002, 2003 IBM Corporation

Agents run inside the JVM

The JVM process sends heap allocation and threads to the profiler agent

The Profiler agent automatically get notified of certain events and can request certain info.

The JVMPI is a two-way function call interface between the JVM and an in-process profiler agent.

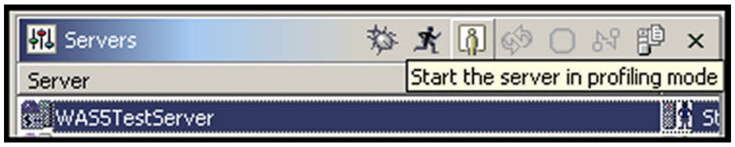
The virtual machine notifies the profiler agent of various events referencing heap allocation, thread start, etc.

On the other hand, the profiler agent issues controls and requests for more information through the JVMPI. For example, the profiler agent can turn on/off a specific event notification, based on the needs of the profiler front-end.

This diagram was taken off of Sun's Website

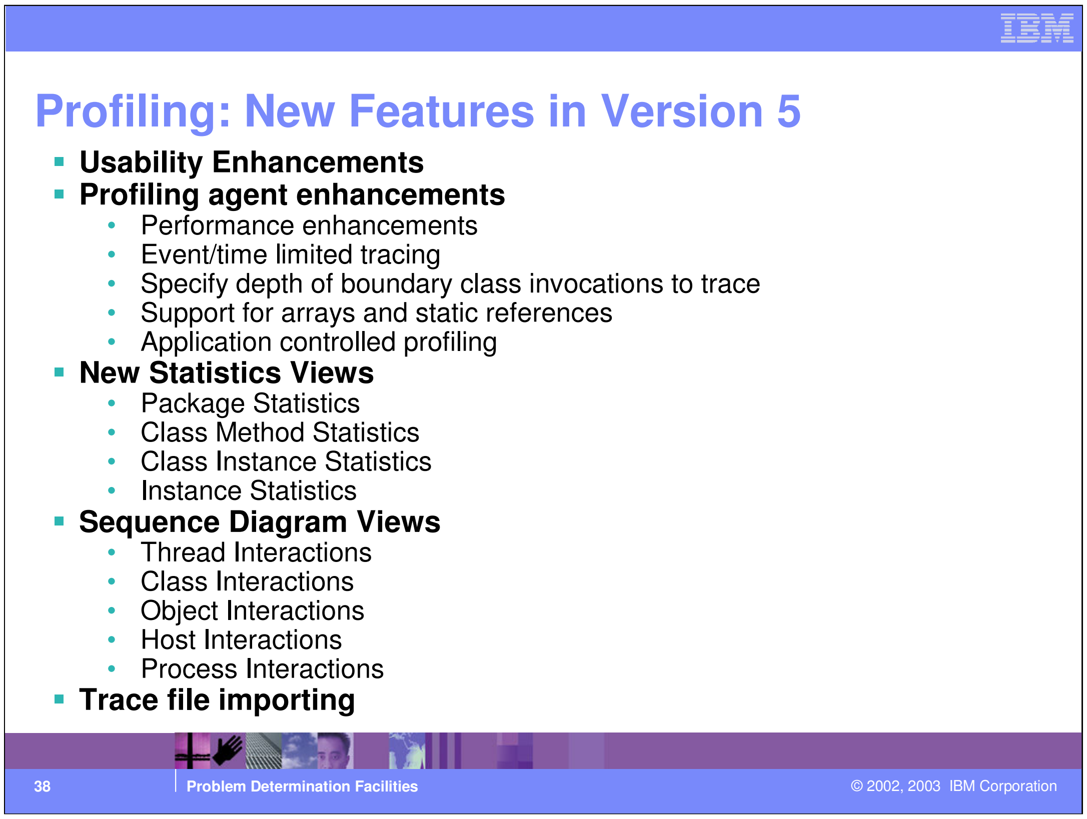
Profiler Agent

- **Gathers information about**
  - Execution flow
  - Performance analysis
- **Gathered information is passed to the Remote Agent Controller**
- **Specify -XrunpiAgent on the JVM command line to invoke the profiling agent manually**
  - Causes JVM to invoke the profiler agent library piAgent
- **Use "Start the server in Profiling mode" button in Servers View to start a server in the WebSphere Test environment in profiling mode**



The Profiler Agent will use JSR-47 which is a specification for logging APIs within the JavaTM platform. These APIs will be suitable for logging events from within the Java platform and from within Java applications. The vision is that it will be possible to enable or disable logging at run-time. It will be possible to control logging at a fairly fine granularity, so that logging can be enabled or disabled for specific functionality.

The logging APIs will allow registration of logging services at run time, so third parties can add new log services. It will be possible to provide bridging services that connect the Java logging APIs to existing logging services (e.g. operating system logs). Where appropriate, the logging APIs will also support displaying high-priority messages to end users.



Profiling: New Features in Version 5

- **Usability Enhancements**
- **Profiling agent enhancements**
  - Performance enhancements
  - Event/time limited tracing
  - Specify depth of boundary class invocations to trace
  - Support for arrays and static references
  - Application controlled profiling
- **New Statistics Views**
  - Package Statistics
  - Class Method Statistics
  - Class Instance Statistics
  - Instance Statistics
- **Sequence Diagram Views**
  - Thread Interactions
  - Class Interactions
  - Object Interactions
  - Host Interactions
  - Process Interactions
- **Trace file importing**

38 Problem Determination Facilities © 2002, 2003 IBM Corporation

Statistics views have been added and revamped to organize information collected by the profiler agent in a more intuitive fashion.

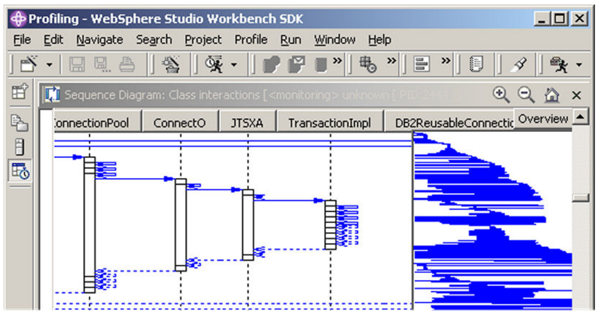
Start by looking at the Packages view and work your way down to the more specific statistics views.

Sequence diagrams are a new feature in WSAD. They allow for the visualization of the execution of Java applications in the form of sequence diagrams that are defined by the Unified Modeling Language (UML) notation. Use to find the hotspot in your program and then use traditional profiling (profiling agent) to dig deeper into the problem.

Trace file importing allows the user to direct the RAC to place its trace data into a file. The file can later be imported into WSAD and analyzed. This is ideal for large trace files. WSAD can be used to bring in only a certain amount of the trace file (ex: first 25 MB, etc.)

Sequence Diagrams

- Visualize the execution of Java apps in the form of UML sequence diagrams
- Shows interactions among objects, classes, hosts, processes, threads, and J2EE components


Profiling - WebSphere Studio Workbench SDK
File Edit Navigate Search Project Profile Run Window Help
Sequence Diagram: Class interactions
onnectionPool
ConnectO
JTSXA
TransactionImpl
DB2ReusableConnectic
Overview

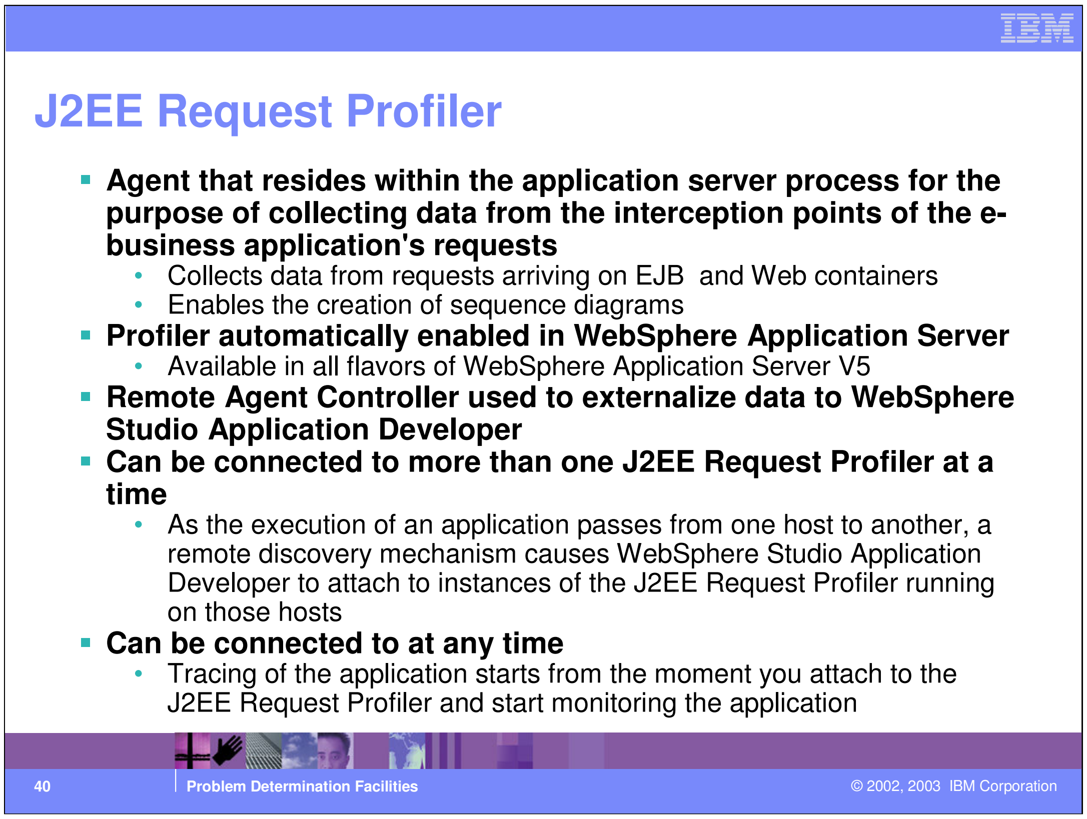
You can use class interaction diagrams to view interactions of class methods that participate in the execution of an application. For applications that execute on a single machine, classes may be viewed from the level of either the host or the process.

To view class interaction diagrams, follow these steps:

Select the monitor that represents the application cluster, host or process of interest in the Monitors view of the Profiling perspective.

From its pop-up menu, select Show View > Class interactions. The Sequence Diagram view shows entities that correspond to classes in the monitor, host or process. Methods of those classes interact by the means of method calls.

In the screen shot notice the Overview view that has been overlaid on the diagram. The overview map allows the user to quickly move around the diagram by double-clicking on a point some where in a map, moving the whole focus of the view. The Overview button is found in all Sequence Diagram Views.



J2EE Request Profiler

- **Agent that resides within the application server process for the purpose of collecting data from the interception points of the e-business application's requests**
  - Collects data from requests arriving on EJB and Web containers
  - Enables the creation of sequence diagrams
- **Profiler automatically enabled in WebSphere Application Server**
  - Available in all flavors of WebSphere Application Server V5
- **Remote Agent Controller used to externalize data to WebSphere Studio Application Developer**
- **Can be connected to more than one J2EE Request Profiler at a time**
  - As the execution of an application passes from one host to another, a remote discovery mechanism causes WebSphere Studio Application Developer to attach to instances of the J2EE Request Profiler running on those hosts
- **Can be connected to at any time**
  - Tracing of the application starts from the moment you attach to the J2EE Request Profiler and start monitoring the application

40 Problem Determination Facilities © 2002, 2003 IBM Corporation

There is only one instance of the J2EE Request Profiler that is active in a process that hosts the WebSphere Application Server.

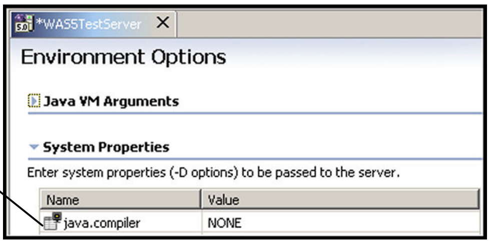
Whenever the J2EE Request Profiler is active in a process, it is active on the host that contains that process as well.

There can be multiple instances of the J2EE Request Profiler that you are attached to.

Remote agent controller (RAC) needs to be installed and active on each machine in order for remote discovery to work.

General Profiling Steps

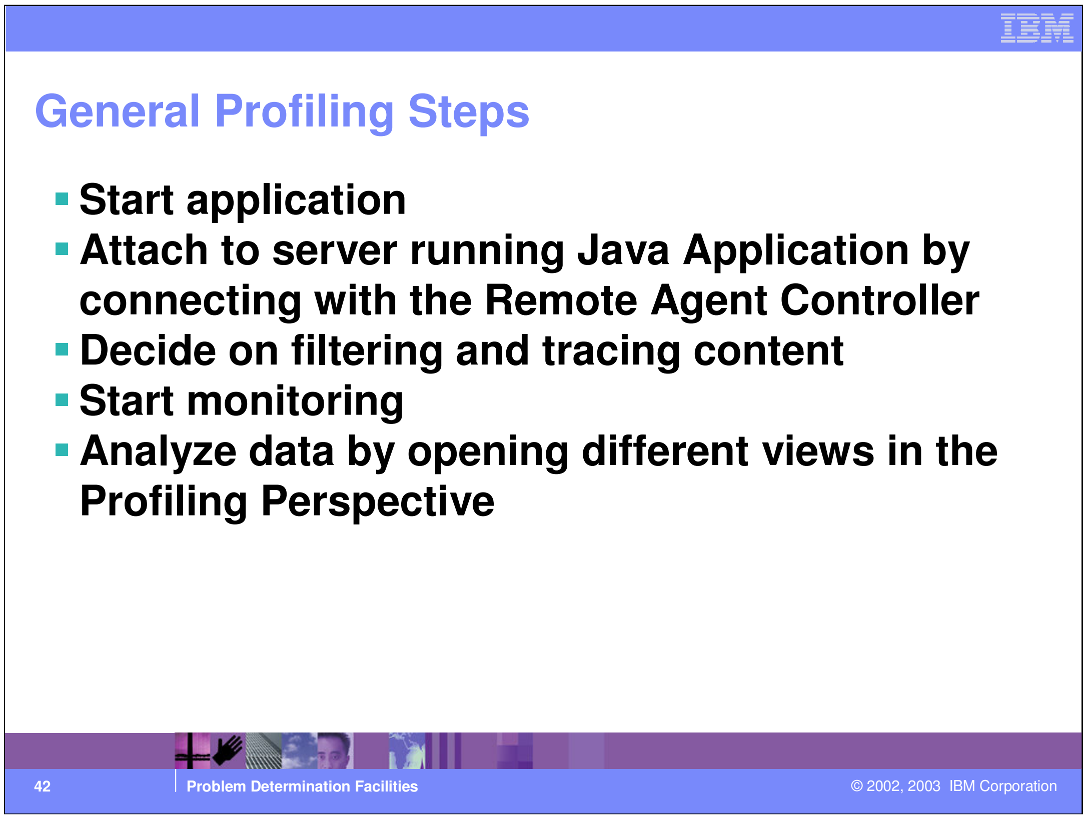
- **Install RAC on the system**
- **Enable PI agent within JVM**
- **Turn JIT (Just In Time) compiler off**
  - •Specify -Djava.compiler=NONE on the JVM command line
  - •In WebSphere 5.0, check the "Disable JIT" checkbox under
  - •In WebSphere 4.0.1, check the "Disable JIT" checkbox under Advanced JVM Settings tab
  - •In WSAD, specify a System property of java.compiler with a value of NONE in the Environment tab of the server configuration



Enter system properties (-D options) to be passed to the server.

Name	Value
java.compiler	NONE

The JIT compiler takes the Java and converts the byte code into machine language at run-time. This has to be disabled in order to profile an application. The Profiler in WebSphere Application Developer needs to examine the code line by line and not the interpreted machine language to get an accurate depiction of the data.

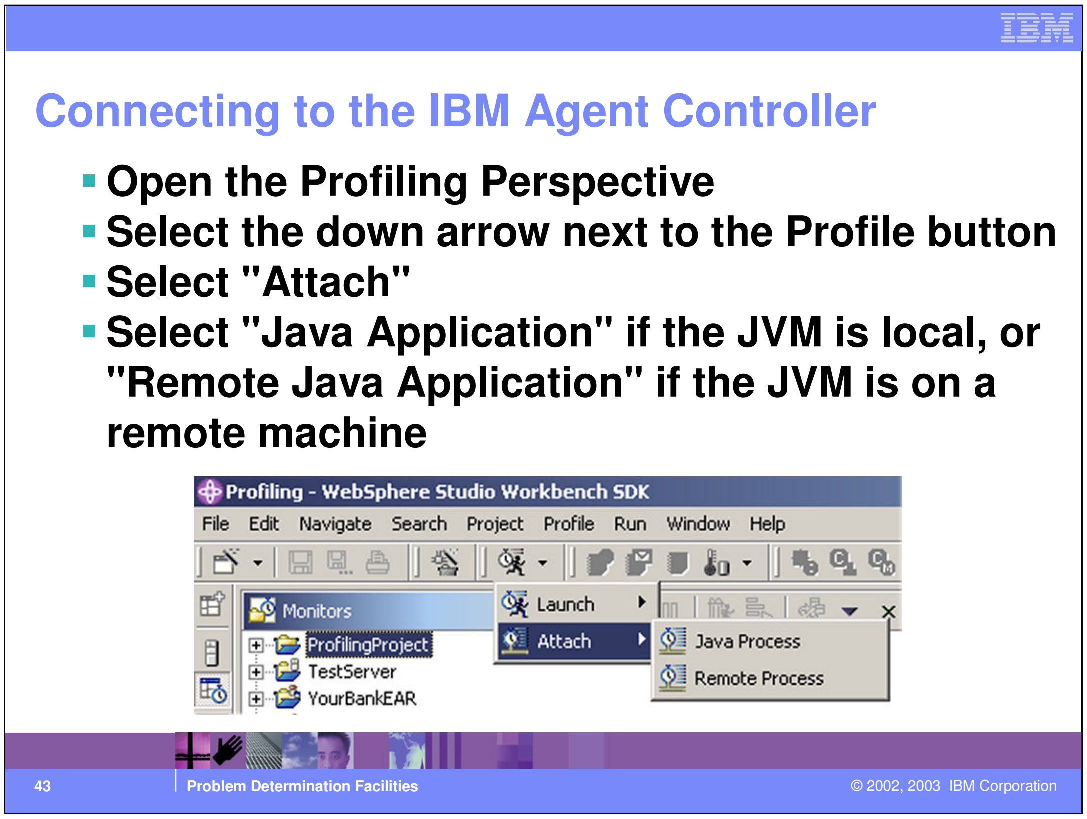


General Profiling Steps

- Start application
- Attach to server running Java Application by connecting with the Remote Agent Controller
- Decide on filtering and tracing content
- Start monitoring
- Analyze data by opening different views in the Profiling Perspective

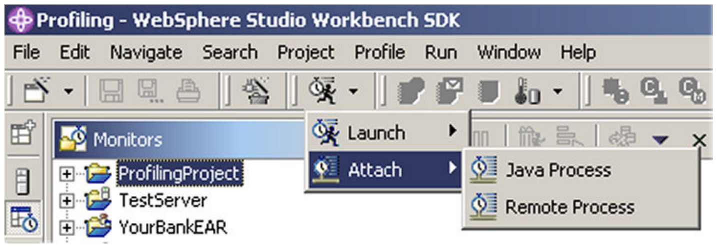
42 | Problem Determination Facilities © 2002, 2003 IBM Corporation

You may have to hit the refresh button to display the current performance data upon opening the different views.



Connecting to the IBM Agent Controller

- Open the Profiling Perspective
- Select the down arrow next to the Profile button
- Select "Attach"
- Select "Java Application" if the JVM is local, or "Remote Java Application" if the JVM is on a remote machine

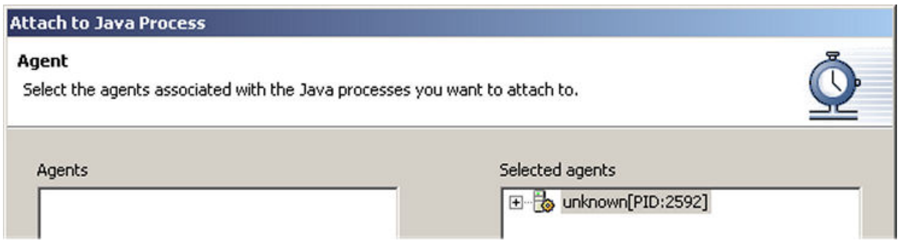

Profiling - WebSphere Studio Workbench SDK
File Edit Navigate Search Project Profile Run Window Help
Monitors
ProfilingProject
TestServer
YourBankEAR
Launch
Attach
Java Process
Remote Process

When you select "Java application" if the JVM is local you will be asked for the machine name and what port to use. If you select Remote Java Application you be asked where the JVM is.

When you attach to an agent in a process, you have to manually start monitoring the agents (select Start Monitoring from the pop-up menu of the agent). However, when you launch a process, monitoring of the agents is started automatically.

Attach to Agent(s)

- **Select the java process to attach to**
  - Process ID should match that of the server you want to monitor (see server's console panel)
- **Decide on filtering content**
- **Decide on profiling options**


Attach to Java Process

Agent

Select the agents associated with the Java processes you want to attach to.

Agents

Selected agents

unknown[PID:2592]

The agent is a piece of software that resides within a host process. An agent may run on both the client and the server.

In the Profiling perspective, it is part of the technology used to profile applications. It collects data about the execution of an application program. The profiling views provide visualizations of the information that is collected by the agent.

Agents can be identified by different symbols, which signify that they are in different states:

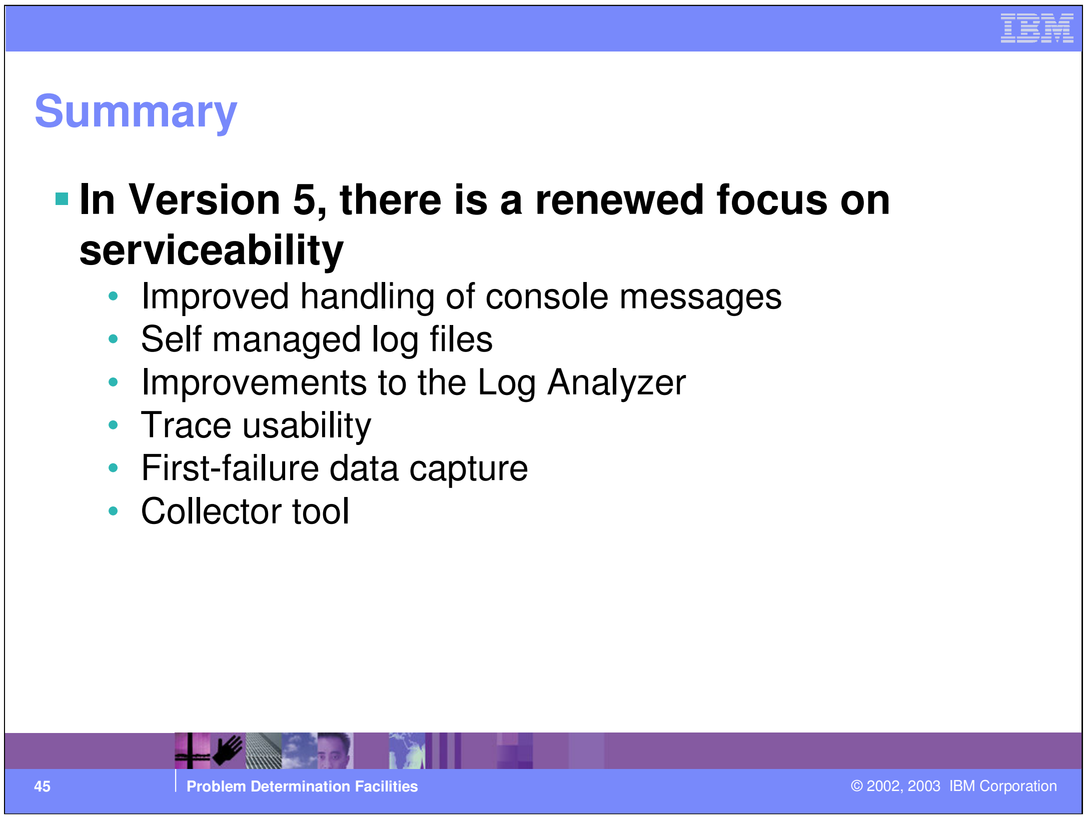
Pause symbol - The agent is running.

Play (triangle) symbol -The agent is running and collecting data.

Stop symbol (agent symbol is white) - The agent is no longer alive. In the case of the profiling agent, which maps to the lifetime of the process, it indicates that the application has run to completion, and the agent has exited or completed its work.

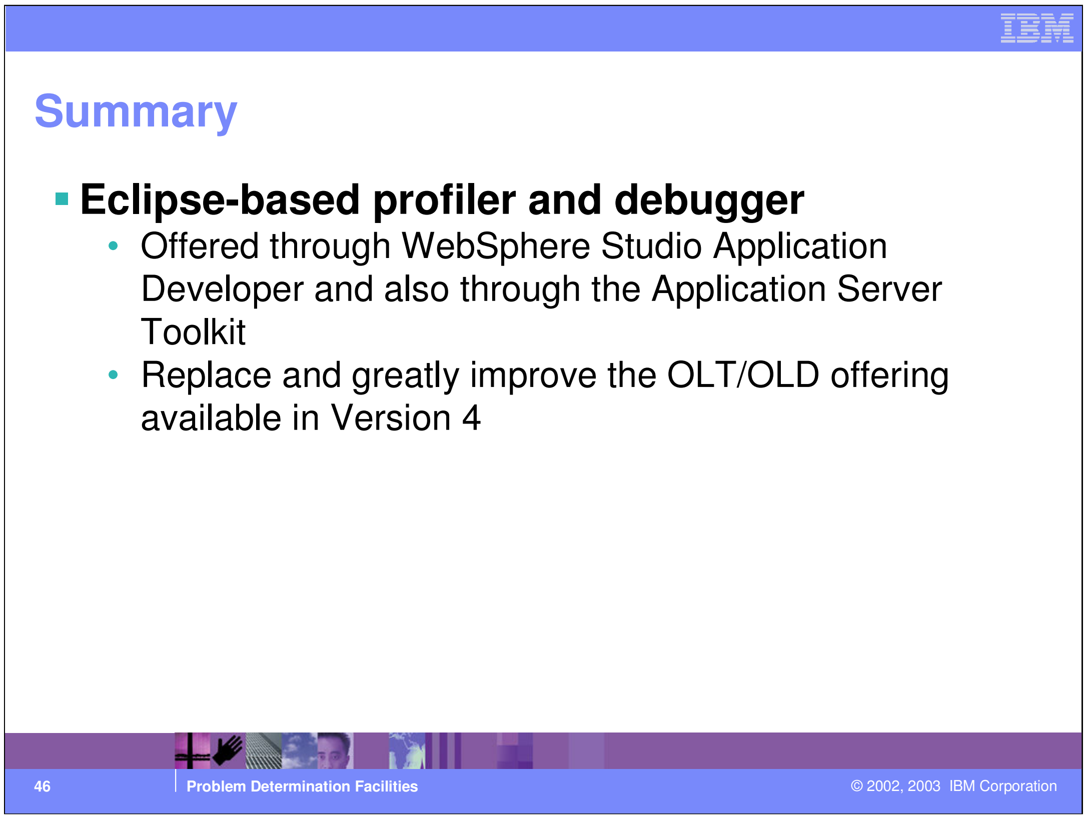
Detached symbol - The process is running, but the current workbench client is not attached to the agent. Any other workbench can attach to this agent, and start monitoring the application.

Note: Each agent can be attached to by only one client at a time.



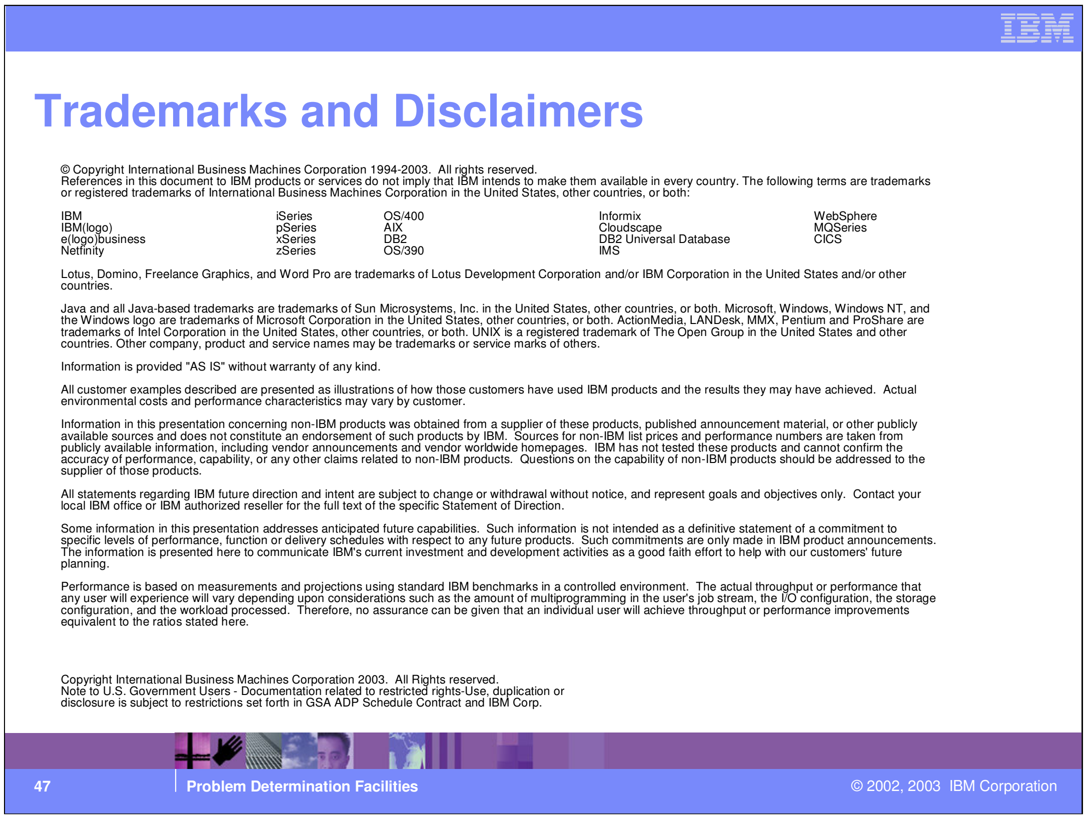
Summary

- **In Version 5, there is a renewed focus on serviceability**
  - Improved handling of console messages
  - Self managed log files
  - Improvements to the Log Analyzer
  - Trace usability
  - First-failure data capture
  - Collector tool



Summary

- **Eclipse-based profiler and debugger**
  - Offered through WebSphere Studio Application Developer and also through the Application Server Toolkit
  - Replace and greatly improve the OLT/OLD offering available in Version 4



Trademarks and Disclaimers