

**IBM Response**  
to the study entitled  
**“Comparing Microsoft .NET and IBM  
WebSphere/J2EE”**

by The Middleware Company

**(Microsoft Response to this document provided inline)**

*October 2004  
IBM Competitive Technology Lab*

## Another Flawed Study

*Microsoft note:*

*As with previous benchmarks Microsoft has sponsored between .NET and J2EE, including previous studies conducted by The Middleware Company (TMC), all of the code and test methodology for the study, as well as tuning parameters and detailed notes have been published online, for full disclosure. This allows vendors and customers to review the artifacts and even replicate the tests on their own to verify the results. This policy has produced commentary on results that otherwise would not be possible.*

*This is in stark contrast to benchmarks quoted by IBM that purport to compare .NET to WebSphere. In these cases, results are quoted, but no code is provided, no details are provided, and there is no chance by third parties to verify or replicate the results. Ultimately, Microsoft encourages customers to download the report and code, read the report, and conduct their own testing and technical comparison to make the most informed decision. At the same time, customers should understand that the current .NET vs. WebSphere study published by TMC is accurate. The notes below constitute our response to IBM's effort to paint the results as inaccurate. The full study is important to read, since only after reading the full study can customers appreciate the attention to detail, and the comprehensive and very balanced nature of the study. The study is available at:*

<http://www.middlewareresearch.com/endeavors/040921IBMDOTNET/endeavor.jsp>

The latest Middleware Company study is flawed and does not accurately reflect the capability of WebSphere J2EE vs. Microsoft .NET. Like two previous discredited Middleware Company studies, this study was funded by Microsoft. While expert Microsoft programmers were allowed to contribute to the .NET side, neither IBM nor other WebSphere J2EE product experts were invited to contribute to the testing.

*Microsoft note:*

- 1. IBM mentions two prior App Server performance studies completed by TMC in 2002 and 2003. Neither one was in the end discredited, although the first one caused quite a bit of controversy and ultimately lead to politics within Middleware since they found .NET outperforming J2EE and enormous pressure was put on the firm by Sun, BEA and IBM to retract their results. In the end, TMC repeated the original study and largely verified the results (more below). In fact, studies by TMC have been so widely read and were so thorough that **IBM themselves recently hired TMC to complete a productivity study of their J2EE development tool, Rational Rapid Developer; this study is published on the Middleware site.***
- 2. Microsoft programmers did not contribute to the .NET effort. As the study report clearly documents, neither developers from Microsoft nor IBM were allowed to contribute by design, as the study was meant to show what a customer team could expect to accomplish without direct involvement from a vendor, using only standard online and phone support. So Microsoft **had no involvement** in implementing the application in the study, and the study report is quite clear on this.*

The Middleware Company publicly retracted the results of its previous J2EE vs. .NET study because of serious errors in its testing methodologies and its failure to invite J2EE vendors to participate.<sup>1</sup>

*Microsoft note:*

*Some history on this situation. The initial performance study done by TMC was published in October 2002. .NET came out on top by such a large margin, that the results were widely believed to be flawed by the J2EE community. In response, TMC conducted a second study, publishing the results in July*

---

<sup>1</sup> In the Appendix is the complete text of The Middleware Company retraction letter published in November 2002.

*2003, which Microsoft agreed to fund. IBM was publicly invited to participate in this second study, and declined. This was noted in the study report, dated July 2003. Of note, IBM has also been invited to participate in WebSphere/.NET performance tests conducted by PC Magazine, eWeek, and Doculabs just to mention a few and has declined in all cases. Microsoft is aware of this because Microsoft was also invited to the same tests, and agreed to participate, only to later learn than IBM had not agreed.*

*Though IBM did not participate directly in the July 2003 study by TMC, numerous J2EE experts did participate, including noted industry analysts and J2EE architects/authors. These experts collaborated to produce a completely new implementation of the benchmark application for the July 2003 study, one vetted by the community for "best practices" approach. See the final report on [www.middlewareresearch.com](http://www.middlewareresearch.com) for details.*

*The results in July 2003 once again showed that .NET had superior performance when compared to the best J2EE offerings. The results for one Java-based application server improved significantly as compared to the 2002 study; however, .NET delivered results better still. And IBM WebSphere delivered significantly worse results than .NET. TMC published these results and has never apologized for them or retracted them.*

*Shortly after these results were published, IBM had claimed the .NET results were not valid for a variety of reasons, among them, the Microsoft implementation had used stored procedures which IBM supposed conferred an unfair performance advantage. However, this was not true. Further tests proved that the use of stored procedures in the application architecture had no significant performance impact one way or the other.*

*The current study is the third large comparative study TMC has conducted. It is focused not only on performance, but also on productivity and cost, including time and labor cost, required to support the various application platforms.*

This study has similar problems. The products and toolsets compared in the testing were mismatched, and relatively crude WebSphere J2EE programming techniques were used rather than the most optimal tools and programming methodologies. Statements made within the study show that the choice of tools and methodologies were purposeful.

*Microsoft note: This is false. The WebSphere team in the current TMC study made their choices based on what they believed, based on their experience building for J2EE app servers including IBM, would give them the best results for productivity, performance, reliability and manageability. Microsoft in no way had any influence over these choices, and TMC stands by these choices. In fact, with the Rational Rapid Developer implementation, since RRD generates the code, there was no real architecture choice to make.*

The cost comparison between IBM and Microsoft also is inaccurate and does not represent a real-world environment. In addition, there were important cost considerations missing on the Microsoft side.

*The detailed commentary below shows this claim to be unfounded. The cost comparison published by TMC is based on published prices for IBM and Microsoft software. The cost differential is significant: this study found WebSphere licenses to be 10 times the cost of .NET licenses inclusive of Windows and Linux license costs.*

The Middleware Company's WebSphere J2EE programmers failed to use development practices that would have yielded more favorable results for WebSphere J2EE—for example, practices that would enable software reuse across enterprise architectures. The study also did not attempt to measure the advantages of team development across a business with multiple team members involved. Rather, the study timed the work of three relatively inexperienced WebSphere developers. The results are misleading in each of the categories the tests aimed to measure--Development Productivity, Tuning Productivity, Performance, Manageability, and Reliability.

*Microsoft note:*

*This was the most exhaustive study of its kind ever completed, taking over 9 months to complete. While IBM may wish they would have gotten better results using different team dynamics, this is just wishful thinking. If a team of three highly skilled J2EE developers took almost 100 days to tune two different IBM WebSphere implementations, including hiring two independent IBM WebSphere consultants to verify their configuration, this is very telling and revealing of just how difficult it is to configure and tune WebSphere. The Middleware notes detail precisely where they encountered the difficulties.*

In its apology and retraction letter for its last J2EE vs. .NET test, The Middleware Company wrote: "We also admit to having made an error in process judgment by inviting Microsoft to participate in this benchmark, but not inviting the J2EE vendors to participate in this benchmark."<sup>2</sup>

Inviting an expert Microsoft partner to participate while not inviting IBM or an IBM partner to participate was once again a fatal flaw. Using outside experts for the Microsoft .NET programming team while using its own employees for the WebSphere J2EE programming team is an obvious conflict of interest in a study funded by Microsoft.

*Microsoft note: the study was designed to simulate the experience of a corporate development team, not teams of developers directly from the vendor. Neither the WebSphere implementation nor the Microsoft implementation was constructed with the participation of a vendor. Customers can download the code to verify the results.*

We believe that if J2EE development best practices had been used, the developer productivity would have exceeded that of Visual Studio.NET. The performance and manageability of the WebSphere J2EE application also would have exceeded the published numbers.

*The tricky question here is what are the best practices for J2EE? In several cases, the WebSphere team was using best practices documented by Sun, and yet IBM claims that they were not best practices. This confusion is typical in the J2EE programming model, and is the source of a programmer productivity tax. One thing these studies have shown over and over again is that with J2EE, often times the huge number of architectural choices, with no clear guidance on the tradeoffs between them, leads to failed implementations and loss of productivity. This study and IBM's response proves that even the J2EE vendors themselves do not agree on how to build a J2EE application. for example, use EJBs or no EJBs? Use stateless session beans or entity beans? Use JDO vs. entity beans? Use POJOs with embedded JDBC? Use a straight servlet architecture instead? Use a third party framework (eg. Struts) or not? The choices are complicated, and many customers find out late in the game they have made the wrong choice. A choice that works well in one application server, may not work well in another.*

Microsoft's track record on these types of studies is not good, and users should generally approach them with caution. Besides two previous Middleware Company studies that were retracted,<sup>3</sup> Forrester Research issued a disclaimer of its Microsoft-sponsored study on the cost of ownership of Linux vs. Windows,<sup>4</sup> and a similar IDC Linux/Windows study was discredited in *Business Week* by one of its authors.<sup>5</sup>

*IBM has so far refused to use WebSphere Application Server in any open, industry-sponsored*

---

<sup>2</sup> The Middleware Company, "A Message to the J2EE and .NET Communities," November 8, 2002.

<sup>3</sup> The initial Middleware Company Pet Shop study and the J2EE vs. .NET study cited in the retraction in the Appendix.

<sup>4</sup> "Forrester Alters Integrity Policy," CNET News.com, October 7, 2003. "Forrester published a new policy that forbids vendors that commission studies from publicizing their results. The new policy follows the release last month of a study that concluded that companies spent less when developing certain programs with the Windows operating system than with Linux. Giga polled just 12 companies for the study, which led some critics to question the report's statistical relevance."

<sup>5</sup> "Pecked by Penguins," *Business Week*, March 3, 2003. Says the report: "One of the study's authors accuses Microsoft of stacking the deck. IDC analyst Dan Kusnetzky says the company selected scenarios that would inevitably be more costly using Linux."

*benchmarks, including TPC-W, and benchmark efforts proposed by various independent publications. IBM has most recently lead an effort to ban the publication of price/performance from all SPEC benchmarks, an effort that was unfortunately successful, since it allows IBM to publish SPEC benchmark data without publishing customer pricing data. IBM was apparently trying to avoid comparisons such as the one TMC published, which show that IBM infrastructure software is much more expensive than Microsoft software. Keep in mind that price/performance metrics are a part of every other mainstream benchmark, including benchmarks conducted by TPC, independent publications, and other vendor-neutral organizations.*

*In the absence of bona-fide participation in open industry benchmarks from some vendors, Microsoft has attempted to fill the void by commissioning independent studies. These do not replace industry benchmarks, but without support from IBM, there will never be any industry benchmark. So these commissioned studies are a best effort replacement. Customers should always perform their own due-diligence and test for themselves. In an effort to encourage this, the full source code of the applications as tested is available from TMC for download. Customers can obtain the code, modify it if they think necessary, and conduct their own tests, to decide for themselves.*

*In summary, the Microsoft track record around performance benchmarks is that we encourage benchmarks and comparative case studies, we support vendor-neutral industry efforts where possible, and we commission independent studies where industry efforts do not exist or are lacking support.*

*IBM's track record is much different. The author of the "IBM Response" to the TMC case study is from the IBM Competitive Technology Team. This IBM team itself has conducted performance benchmarks, under the names Trade2 and Trade3, and has selectively disclosed results to customers, press, and analysts. These results are never publicly disclosed, because they employ code that Microsoft developed and gave to IBM for the purposes of interoperability, with the explicit understanding that the Trade2 code would not be used for benchmarking. Despite this understanding, the IBM CTT uses the Trade2 code as "exemplary .NET application" and uses performance results measured on Trade2 as evidence that WebSphere outperforms .NET. But customers and independent reviewers are never able to see the code, to evaluate it for themselves.*

*So, here in this statement, once again, Microsoft would like to publicly encourage IBM to participate in open, industry-supported benchmarks.*

*Having said all of this, performance was only one piece of the current case study report. The report also measured the time and labor cost required to build and operate an application, according to a provided specification, as well as the reliability and operational behavior of an application under load.*

A closer look at how the tests were conducted and the results framed reveals why The Middleware Company study is without merit.

*Microsoft does not believe the report from TMC is the "last word" on the application platform decision. However, likewise it is not "without merit" as IBM argues. This study is one data point, an addition to the discussion. Microsoft believes it provides some convincing evidence that the Microsoft platform, consisting of Visual Studio for tools, Windows Server for running applications, and .NET as the programming model, is far superior to WebSphere Application Server on Linux, with J2EE as the programming model. The evidence shows that in acquisition costs, developer productivity, administrator and operator productivity, and other areas, the Microsoft platform is superior.*

*However, this is just one case study, and each enterprise brings different requirements. Customers should evaluate for themselves.*

## **Wrong "WebSphere" Tool**

One of the largest faults of the study is the focus on Rational Rapid Developer (RRD) as the primary J2EE development tool from IBM. The results for RRD are represented as "WebSphere" and RRD is characterized as the main WebSphere development tool. However, IBM has never marketed RRD as a full-blown WebSphere

toolset with all the functionality of WebSphere Studio Application Developer (WSAD). WSAD is the tool intended specifically for building integrated enterprise J2EE applications.

*Microsoft note: In fact, IBM had contracted with TMC previously to publish a report on the benefits of RRD. At that time, TMC reported that RRD provided significant programmer productivity advantages for developers, as compared to mainstream tools such as WSAD. This is why RRD was selected for this case study as well. And the J2EE Middleware team also developed a second WebSphere implementation using WebSphere Studio App Developer (WSAD), to compare the results across these two J2EE IBM tools.*

Comparing VS.NET to RRD is not a fair comparison, nor is it the comparison IBM would have recommended. Why then did The Middleware Company characterize Rational Rapid Developer as the primary tool for WebSphere J2EE environment? The choice is particularly odd in light of the fact that they were aware that WSAD was the “mainstream” tool for WebSphere development (and that they achieved better results with WSAD). Articles in the trade press reported that the WSAD results were “unofficial” because they were not as controlled as the RRD measurements. This explanation was provided by The Middleware Company and Microsoft.

The WSAD results in this study were comparable to VS.NET in developer time spent on the project. The approach to application development in RRD is very different from WSAD, so the theory postulated in the study that the WSAD results were boosted by the experience the team gained using RRD are not credible.

*Microsoft note: Why is it not credible, that during the 2<sup>nd</sup> try at implementing a specification, the team will have not benefited from learnings they achieved in the first try? Clearly, the team knew the ins and outs of the specification much better, having implemented it fully, once.*

The bottom line is that The Middleware Company chose to emphasize and give more weight to a tool that would not be recommended by IBM as the primary tool for WebSphere J2EE development. By focusing on RRD, the study draws attention away from the more favorable results posted by WSAD, which is IBM's premier application development tool and the direct competitor to VS.NET. Representing RRD in this way is inaccurate and misleading.

*Microsoft note: Well it's certainly interesting to note IBM is now telling customers that its primary 'rapid developer' tool is not suitable for WebSphere development. This seems to be contrary to previous statements from IBM. As for the WSAD development, the Middleware team found RRD results to be so poor that all parties agreed that an additional WSAD implementation had to be created to more thoroughly measure WebSphere results. The study notes the outcome. The results still did not match the .NET results. Customers can fairly easily repeat this portion of the test themselves to verify.*

The table below summarizes the development tool comparisons made in the study:

Development Productivity	.NET beats RRD	.NET = WSAD	WSAD beats RRD
Tuning Productivity	.NET beats RRD	.NET = WSAD	“uncertain”
Performance	.NET beats RRD	.NET = WSAD	WSAD beats RRD
Manageability	.NET beats RRD	.NET beats WSAD	RRD beats WSAD

RRD = Rational Rapid Developer  
 WSAD = WebSphere Studio Application Developer

In order to have a fair comparison, you should make more of an attempt at having a level playing field. Here are the specific problems in this study which invalidate the results.

## Unequal Programming Teams

*Microsoft note: Middleware stands by the results of the study. The two teams were similarly skilled in terms of experience as the study report notes. The WebSphere team even brought in two independent IBM/Linux certified consultants to verify their configuration.*

Among the most questionable claims made in the study were that the .NET and J2EE programming teams were equal in programming experience and knowledge of their respective platforms. Having programming teams of equal skills is critical to the credibility of the end results.

The report states that three Microsoft “senior developers” were sourced from a Microsoft Solution Provider (Vertigo Software). The J2EE developers were sourced from The Middleware Company. Why were the J2EE developers not sourced from an IBM business partner that would have had some knowledge about IBM products?

There is an obvious disparity in the skill level of the .NET and WebSphere team exhibited during the actual development work. The Microsoft team exhibits amazing experience and depth of knowledge. An example is the Microsoft team’s decision to code distributed transactions using “ServiceDomain classes” instead of the “COM+ catalog.” ServiceDomain classes are very new, very complex, and not that well understood and documented.

*Microsoft note: Not true. ServiceDomain was introduced with .NET V1.1, which is one year old. They are a simpler way of using transactions, and many many examples are available. They are well understood by the community, and well documented. Perhaps they are not well understood by IBM.*

Another example of the high skill level of the .NET developers was their use of “Data Access Application Block” (DAAB) for enhancement of performance. This is not a standard part of .NET and one must go out explicitly and download and install it. To decide to code the distributed transactions with ServiceDomain classes and to add DAAB for performance enhancement, and to have it work the first time, shows a deep amount of technical skill.

*Microsoft note:*

*Again, Not true. The DAAB is part of the patterns and practices Microsoft publishes and encourages all .NET architects to learn and use. DAAB has been available for over two years, and over one hundred thousand developers have downloaded and used it for that period of time. Microsoft publishes these “Application Blocks” as re-usable, tested, proven blocks of code that can be dropped into existing projects. This is a best practice when building in .NET, and the use of Application Blocks has been encouraged and endorsed by Microsoft for years. IBM suggests that re-using these proven and tested blocks of code implies a highly skilled developer. On the contrary, the Application Blocks actually do the reverse. They encapsulate the skills and knowledge of skilled developers so as to lower the barriers to producing high-quality, high-performance reliable .NET applications for non-expert developers.*

The WebSphere team, on the other hand, exhibited little or no experience with WebSphere Studio Application Developer. Their inexperience is obvious and is easily traced throughout the study:

- The study reports (pg.37) that “the .NET team was undoubtedly more experienced with their tool than the J2EE team with theirs.”
- The J2EE team decided to install the Sun JVM rather than the IBM-supplied JVM 1.4.1. WebSphere Studio product requirements state to use the IBM JVM or unpredictable results can occur. The unpredictable results that occurred later (pg. 59-60) were an unrecognized parameter during garbage collection. The time it took to troubleshoot and fix the problem counted against the J2EE team. The substitution of the Sun JVM can only be explained lack of experience with WebSphere Studio. If the intent was to test how WebSphere Studio performed, why would you introduce a non-IBM JVM?
- The J2EE team decided not to do any source code management (pg. 32), even though ClearCase LT and CVS were available. “Instead they divided their work carefully and copied changed source files between their machines.” The .NET team used Visual SourceSafe for source control. Does the J2EE decision indicate experienced developers? Problems resulted (pg. 45) for the J2EE developers that increased their development time. It appears again that lack of experience played a role in this poor decision.
- The J2EE team decided to use the Sun J2ME wireless toolkit in Sun One Studio to create an application for a mobile device that was part of the specification. The code generated from the Sun wireless toolkit did not work with

the RRD-generated application. Again, this displays a lack of experience in not using the WebSphere Studio tools to interface with the mobile device.

- It took the J2EE team extra time to discover that they had a SIMPLE rights (SELECT rights) user error when they went to access the Oracle database. This counted as developer productivity time and again showed lack of experience.
- The J2EE developer team used POJO (Plain Old Java Object) instead of what best practice might have indicated here – EJBs. It was noted later in the report that the team created a “framework” to handle the JDBC access. If they had used EJB’s, they would have had the container handle the pooled requests and it can be argued that the performance would have been better. In the POJO scenario, a new connection had to be created for EVERY call to the database which is very expensive. In addition, they spent extra time developing a “crude” framework to handle the requests, which counted against them.

*Microsoft note: This seems to be a drawback of the JDBC 2.0 architecture – database connection pooling is not supported. In contrast, the .NET managed provider for SQL Server automatically creates and manages database connection pools, without any extra effort by the developer. It is true that Sun aims to rectify this shortcoming in upcoming JDBC specifications, but the current WebSphere implementation does not provide auto-pools for database connections.*

- The J2EE team chose not to use (or did not seem aware of) the more productive tools available such as Java Server Faces and Service Data Objects.

*Microsoft note: Yes, this seems to be characteristic of Java-based platforms. There are many options and not every developer is well educated on the strengths and weaknesses of them all, with respect to each other.*

- The J2EE team specifically chose not to use the Model View Controller approach such as STRUTS technology.

*Microsoft note: The many choices faced by the WebSphere development team, and their tendency to make what IBM characterizes as the “wrong choices”, only underscores the complexity of the WebSphere environment and the prevalence of the pitfalls facing all developers. Conversely, the practices followed by the .NET team are well-known, mainstream approaches. This is one of the primary benefits of .NET – simplifying development and removing pitfalls, so that the simple things are simple, and the hard things are possible. In WebSphere, the simple things are difficult, and the hard things are impossible.*

---

*Microsoft note: In summary, all of these choices are reasonable and accurately reflect some of the many choices IBM WebSphere customers must make as well when conducting a WebSphere project. They do not prove that the WebSphere team lacked experience, but rather demonstrate the exploding matrix of choices J2EE and WebSphere force on developers, ultimately making the development process less productive.*

Despite the fact that the three J2EE assigned developers from The Middleware Company had little or no experience with WebSphere Studio, and did not use recommended best practices, **WebSphere Studio was still robust enough to allow the J2EE team to equal the results of the expert Microsoft team using VS.NET in the areas of developer productivity, configuration, tuning and performance.**

*Microsoft note: The results of the effort using WebSphere Studio should be considered in light of a few relevant facts:*

- *In order to maintain a level playing field, the specifications for the application were given to two teams at the same time. One team would use RRD, the other Visual Studio (VS). All the time these teams spent on building the applications was documented and audited. The results show that it is much more productive to build an application in VS, as compared to RRD.*
- *The results were so strongly in favor of VS that a second development effort on WebSphere*

*was undertaken. This time the RRD team, having seen the specification and already produced a result, re-built the application again, using WSAD. To be clear: the WebSphere team had the advantage of already having become familiar with the application specification and the challenges it presented.*

- *Secondly, the time spent on the second effort was not documented and audited. There was no strict control or measurement of the effort.*

*Therefore, claims that WebSphere Studio was able to equal Visual Studio in terms of productivity, performance and other metrics, are to be taken with a large grain of salt. Customers should ultimately build a simple application in each, and decide for themselves. This Middleware study shows Visual Studio.NET offers significant productivity gains over both IBM RRD and IBM WSAD.*

*Furthermore, the WebSphere team took 50% more time than the .NET team to tune and configure their WSAD implementation, on top of the 75 days of additional tuning and configuration work they re-used from the RRD implementation. Any way you look at it, .NET and VS.NET were over twice as productive from development standpoint, and over 4-5 times as productive from tuning and configuration standpoint. This is all clearly documented in the report.*

## Software Platform Disparity

For a fair performance comparison, the software platforms being tested should be equal. However, in addition to the lack of J2EE developer experience, different databases and operating systems were used in this test. Consider the following:

- Although an attempt was made to make the hardware equivalent, that did not hold true for the base development systems. The Microsoft development team's system consisted of Windows XP and a pre-loaded SQL Server database. The J2EE team's system had Windows XP and a pre-loaded Oracle database. These two different databases were used in the production environment as well. Why were two different databases used by the two teams? At the very least, the databases should have been tuned, optimized, tested, and certified that they were performing equally. We have no information or data to indicate how they compared.

**This flaw by itself should be enough to call the entire performance results into question. Before a single line of code was written, the entire study was suspect.**

*Microsoft note: Each team was allowed to choose either SQL Server or Oracle. The .NET team chose SQL Server, and the Middleware team chose Oracle. Oracle and SQL are typical, representative databases for the respective application platforms. Also, in contrast to IBM's statement that there was "no information" on how the databases compared, the auditor's report notes that in all performance tests, neither database exceeded 50% utilization. The database was never the bottleneck. Much time was spent by the WebSphere team in tuning the Oracle database; they had an expert Oracle developer involved in this.*

- Linux was mandated to run the performance tests. A real performance test should compare one competitor to another with as much commonality as possible. The WebSphere J2EE tests should have been run on the same platform (Windows Server 2003) as the .NET tests.

*Microsoft note: This test used the OS platform IBM's most strongly recommends. From IBM's response here, are we to conclude that WebSphere runs better on Windows than on Linux? This is a very, very interesting comment from IBM! Certainly it's fair to compare .NET on Windows Server 2003 to WebSphere on Linux, IBM's primary recommended platform for WebSphere on Intel-based equipment. For IBM to say otherwise is startling. Perhaps WebSphere would have performed better on Windows Server 2003—the great thing is with all the code published, customers can install on Windows Server 2003 and try it for themselves.*

## Single-Tier Restrictions

The configurations in The Middleware Company study were restricted, which benefited Microsoft in performance vs. the J2EE application. The study did not allow for multiple-tier implementations—configurations that would mirror real-world usage. Customers like to separate physical tiers of Web applications because it gives them more flexibility for allocation of resources on a tier-by-tier basis and the ability to separate tiers for security and/or geographical concerns.

*Microsoft note: Each implementation was correctly developed as a multi-tier implementation, and then deployed to application servers according to well-understood, reasonable and customer-recommended best-practices. Adding more physical tiers to the application would be an option, but quite unnecessary in this case. Multi-tier implementations are, in fact, are often not recommended for performance reasons. The advice from IBM, Microsoft, and other vendors is the same: for highest performance, avoid network traversal when possible. Some real-world application configurations do not follow this advice, which gives greater flexibility but inferior performance.*

*This particular benchmark test specified running the separate application layers on a single a consolidated server, for maximum performance. By all measures, this is a typical, best-practice configuration, and it should be a fair and non-discriminatory requirement.*

There were a number of other factors in The Middleware Company study that impeded the J2EE application's performance:

- The Edge server is not needed and adds another layer that is not required. Using the IBM HTTP Server where the Edge Server is, and deleting the two IBM HTTP Servers from the Application Server machine would suffice. This would decrease the complexity and total path length of the application, and allow more processing power for the application servers.

*Microsoft note: Once again, this illustrates the complexity of the WebSphere environment and the many pitfalls that await architects who use WebSphere. When to use Edge Server for load sharing and when to use IBM's HTTP Server?*

- The teams used different patterns to design the applications. The Microsoft team used a stored procedures pattern while the J2EE team did not. It is quite possible that better performance had been achieved on the J2EE side if it had had the same playing field.

*Microsoft note: Again, this has been repeatedly shown by TMC 2003 and other performance tests to be untrue. The query processors in Oracle and SQL Server both can optimize a parameterized query in SQL to deliver equivalent performance to a stored procedure. This is documented in the TMC report. It should also be pointed out that the WebSphere WSAD implementation also used stored procedures for parts of its implementation.*

- Heap size affects the amount of memory allocated to the Java Virtual Machine. One would want to dedicate as much memory as possible to the heap size. The heap sizes were allocated stating that the servers had 1 GB of memory each. This appears to be incorrect. The diagram on page 26 states the servers are 2 GB each, as does the CN2 Technology auditor report. A higher heap size should have been used, something like a minimum heap size of 1,256 MB and a maximum of 1,536 MB.

*Microsoft note: This discrepancy is worth investigating. However, the current statement IBM is offering regarding heap size in the JVM is different than its previously published guidance. Previously, IBM has said that an overly large heap size would cause "Stop the world" garbage collection events, which very adversely affected performance in WebSphere applications. Selecting the proper heap size was a delicate exercise in tuning.*

- The standard method of storing data in the HTTP sessions was not used. The standard method of storing session data for both should have been used.
- The use of two datasources in the J2EE case affects performance negatively. The two datasources allocated were one, an XA compliant for two-phase commit; the other one was not XA compliant. Their claim was that by splitting the requests to each of the data sources, performance would improve. In theory that might be true; however, it basically precludes sharing of data connections between the two datasources, and all the database requests going through the XA datasource would lock the tables for other transactions. If they had a single XA datasource, they could satisfy the two-phase commit and the connection pooling. Performance might have improved because of the ability to cache the connection pools.

*Microsoft note: Bi-Modal Data Access is another J2EE best-practice (aka "Blueprint") documented and endorsed by Sun. Sun later renamed this pattern to "Fast Lane Reader". See <http://java.sun.com/blueprints/patterns/FastLaneReader.html>. Sun says: By using the Fast Lane Reader pattern, [applications] can improve performance by avoiding using enterprise beans. This is precisely the pattern the WebSphere team followed. Is it a good idea, as stated by Sun, or a bad idea, as maintained by IBM here. Who knows? Again this shows the many pitfalls that await a J2EE/WebSphere developer. Which advice is good advice? Which patterns are good to follow, and which are not? Even J2EE veterans are not clear and do not agree on which way is best.*

- In general, the performance runs were done based on achieving a maximum response time of one second. The servers were not driven to maximum capacity. It is never stated what the limiting resource was on the system. A more realistic view of performance and stability is a stress run, where there is no "think time" or wait time between interactions, and the only constrained resource is the application server under test.

*Microsoft note: Yes, the performance took the approach, common in many performance benchmarks, to measure the number of requests that can be served, with the average response time remaining under some specified threshold. In this case, the threshold was 1 second. The well-known TPC-C benchmark is constrained in a similar way. This response time requirement adds a real-world constraint to the benchmark. If the server cannot respond in a timely fashion, then it should not be considered to be "handling the load" adequately. This is a non-discriminatory constraint, common practice in benchmarking, and highly recommended for real-world evaluations.*

- On the integrated scenario, the testers conveniently chose to implement 75% of the application where Microsoft is slightly better than IBM, and only 25% of the application where IBM is better than Microsoft on a 2 to 1 ratio, when run together. If one does the math on the separate scenarios, IBM should have been 650 vs. 550 for Microsoft, yet with integrated scenario IBM is 482 vs. 855 for Microsoft. There are no details on the throughput for each application in the integrated scenario, and there is no attempt to ascertain the limiting resource(s) during the runs, yet we are expected to believe that the applications themselves ran slower because they were run concurrently on separate servers.

*Microsoft note: There is no conspiracy to discriminate against IBM. The test configuration was specified independently, before any tests were run, and before any results had been measured. This criticism does make a valid point, though: every enterprise has a distinct set of requirements. The "integrated scenario" in this case study may not reflect the requirements of every enterprise. Therefore enterprises should evaluate and test for themselves. As for the 75% to 25% mix point, it should be noted that the system that had 75% of the load was a cluster of two servers; whereas the system with 25% of the load was a single server. That is why the load was spread out in this fashion, showing again that IBM is simply throwing darts at the wall.*

---

*Microsoft note: The WebSphere team simply followed IBM's published documents in choosing what software and setup to use, just as the .NET team followed guidance published by Microsoft. That was*

*the point of the study: simulate what a customer might expect to achieve without any direct involvement from the vendor. After reviewing the study, Microsoft experts have found numerous areas on the .NET side too that could have achieved better performance. For example, different thread settings, different queue settings, registry settings, and so on. But in both cases, the development teams did not have the advantage of direct vendor involvement, and since time was being measured, they did not have unlimited time to 'try everything.' The study reflects the real world.*

## Cost Manipulations

The study's cost assessment is inaccurate—adding costs to the IBM side while excluding costs that should have been associated to Microsoft. One such extraneous cost was the use of high-end 4-way servers, which was unnecessary and raised the WebSphere configuration cost significantly. For example, a typical configuration like the one in the study could have been deployed on three 2-way servers instead of the 4-way servers that were used.

*Microsoft note: This complaint is unfounded. The test platform chosen was a 4-way server, which is typical for high-volume web application servers. Using a 2-way server would also have been fine. How is the choice of a 4-way machine "unfair" to WebSphere? Interestingly, the costs are not being disputed by IBM, as they come from IBM's own pricebook. Instead, IBM is saying that customers who wanting to reduce WebSphere costs should run on systems with fewer processors. Customers should raise their eyebrows at this statement, it cuts straight to the truth: WebSphere is much more expensive than Windows Server 2003 with .NET. There is no getting around this fact.*

The Middleware Company's J2EE team initially used a three-node configuration (pg. 28) but added a fourth server to run WebSphere MQ Series. The developers' lack of experience added unnecessary cost to the configuration because a JMS server ships native with the WebSphere Application server at no extra cost. This unnecessary use of hardware and software increased the cost of the WebSphere configuration by more than \$200,000.

*Microsoft note: For clarify, it is not true that adding a single node for WebSphere MQ increased the cost of the IBM configuration by \$200,000. Perhaps this is not what IBM had intended to imply. Also, IBM's statement that the lack of experience of the WebSphere team led to the use of a dedicated message queue server is incorrect. The specification mandated the use of the fourth server as a 'dedicated queue server' such that messages were sent to and stored in a dedicated separate machine. This was a requirement of the specification from the beginning, reflecting the realistic customer deployment scenario where a dedicated queue server is used to process messages. Of course, the .NET implementation similarly used a dedicated queue server to meet the specification.*

While IBM's costs were inflated, we believe the Microsoft configuration costs were understated. The Microsoft configuration was missing an extra year of Software Assurance for the Windows Server 2003 Enterprise Server license. This was due to The Middleware Company's misinterpretation of the terms of the Open Value Agreement, which is three years, not two. The Microsoft configuration should be increased by \$595 per server to cover the cost of an extra year of software assurance.

*Microsoft note: Careful readers of the report will note that the costs included one (1) year of support and maintenance. The IBM configuration included a single year of support - not 2 years, and not 3 years. The Microsoft configuration used Software Assurance (SA), which is 2 years of support, more than is required to satisfy the requirement for 1 year of support. Even adding the \$595 per server, the cost differential still shows that WebSphere is 10 times as expensive to license as the Microsoft platform. The \$595 per server per year, is not going to change this fact.*

In a typical configuration, users are required to authenticate before gaining access to Web applications. Each client device would require a Microsoft client access license (CAL). Client access licenses are fees Microsoft charges customers for each user or device that accesses a Microsoft Windows server and Microsoft's

middleware server. These fees are costly and are in addition to the base server pricing. In Microsoft's Software Assurance licensing scheme, users pay an additional 25% of the CAL cost per year for three years. The WebSphere configuration doesn't require users to pay an extra cost for client access and also allows unlimited customer and external customer access to the middleware servers. As shown in the charts below, the inclusion of CAL cost increases the overall price of the Microsoft configuration.

*Microsoft note: This is incorrect. It is true that CALs are client access licenses, and that the inclusion of these licenses will increase the cost of a Microsoft configuration. It is also true that WebSphere licensing does not include the idea of a CAL. However, the terms under which CALs are required for Windows are not as simple as IBM makes them out to be. The report from TMC states that CALs are not required in this case because the users and devices are not authenticating against the Active Directory within Windows Server. This is not correct. The External Connector license is required when external connections are being made.*

*Also, the pricing for the Microsoft configuration given in the TMC report is incorrect in another way. Windows Server 2003 Enterprise Edition is NOT required for 4-cpu servers. In fact the lower-cost Windows Server 2003 Standard Edition can be run on 4-cpu servers. A revised actual price quote for this configuration is given here:*

Microsoft – Actual Reseller Pricing Quote				
Item	Product	Price/unit	Units	Extended Price
Base Server OS	Windows Server 2003 Standard Edition	\$1100.53 / system	4 servers	\$4402.12
Application Server Function	Included in Windows Server	--	--	--
Unlimited Connections	External Connector	\$3062.81/system	4 servers	\$12,251.24
Developer tool	MSDN Enterprise Subscription	\$2483.99 / seat, per year	2 seats	\$4967.98
Media	CD kit for Windows Server 2003 Enterprise Edition	\$23 / kit	1	\$23.00
Total				\$21,644.34

*The addition of the External Connector licenses and the swap of Windows Server 2003 Enterprise Edition for Windows Server 2003 Standard Edition results in a total actual cost for the Microsoft configuration of \$21644.34. The WebSphere cost remains nearly \$254,000. With these changes in the Microsoft configuration, the cost multiple is still over 10x.*

Another omission was the MSDN enterprise subscription cost that reflected only one year rather than the three-year terms of the Open Value Agreement. This would add an additional \$5,000 to the Microsoft pricing.

*Microsoft note: As stated previously, the pricing configuration was for 1 year of maintenance and support. The 1-year MSDN subscription would satisfy that requirement. Similarly, the IBM WebSphere*

*pricing configuration did not include maintenance charges for a second and third year. WebSphere's terms call for undiscounted maintenance charges of \$3210 per processor, per year, totaling over \$50,000 before discounts, which greatly outweighs the additional \$5000 in subscription charges for developer tools for the Microsoft side. However, as the pricing configuration was for 1 year of support and maintenance, these extra charges were not included, on either side.*

The study's assessment of pricing is skewed in other ways that favor Microsoft. For example, the Microsoft configuration is based on a nonsecure environment that would be considered a security risk in a real-world deployment. The WebSphere architecture provides native reliability and security as a standard and does not require users to pay an extra cost for secure client access.

*This is vague. The Microsoft configuration is not "nonsecure". Customers do not have to pay extra for secure access.*

The Microsoft configuration does not require users to authenticate against Active Directory, allowing anonymous users access to the studies' Web applications. Not using authentication or security simplified the development of the .NET application and left the environment open to the types of security risks that dominate the news headlines.

*In the application specifications, directory-based authentication was not included, authentication was performed against a RDMSB storing usernames and passwords (a typical mechanism used by many, many customers). The presumption for this configuration is that authentication would be provided externally. If however, authentication against Active Directory had been required by the specification, the additional External Connector license, mentioned in the TMC report, would have been required. This would represent \$2000 per server (estimated retail price), for a total of \$8000 additional license fees. At the same time, if LDAP authentication were required by the specification, the WebSphere configuration would have had to include a Tivoli Directory Server or equivalent, bringing in additional costs on that configuration. Tivoli Directory Server is licensed for \$10,700/cpu under Passport Advantage, which would bring in at least an additional \$42,800 into the IBM configuration.*

The Middleware Company's cost model understated the full Microsoft costs required for the scenario. A more thorough study of the total costs of ownership shows that Linux and WAS Express have costs slightly lower than the Microsoft solution. This is in strong contrast to their claim that WebSphere would cost ten times more than the Microsoft solution. Below is our assessment of IBM and Microsoft pricing based on a typical customer deployment.

*It is understandable that IBM would like to use WAS Express for comparisons, as it is significantly less costly than the mainstream WebSphere Application Server. However, WAS Express was not sufficient for this case study for at least 2 reasons. First, WAS Express is not licensed for use on server machines with greater than 2 CPUs. Since this configuration utilized 4-way servers, WAS Express did not fit. Secondly, the specification called for load balancing and failover, a typical requirement in an enterprise. Neither of these capabilities is supported in the lower-cost WAS Express. For both of these reasons, the higher-cost WAS ND was required for the configuration. The cost comparison included in the TMC report, showing that the WebSphere configuration was 10x the cost of the Microsoft configuration, remains valid.*

### WebSphere Configuration with a Secure Customer Environment

Item	Product	Price/Unit		Units		Years of Subscription		3 Year Total Pricing	Annual Cost
Base Server OS	Red Hat Enterprise Linux AS Standard Subscription/ per server pricing	\$1,499	x	3 servers	x	3	=	\$13,491.00	\$4,497.00

Application Server Function	WebSphere Application Server Express/ per processor pricing	\$2,000	x	2 processors per server; 3 servers	x	n/a	=	\$12,000.00	\$4,000.00
Annual Maintenance Agreement	WebSphere Application Server Express/ per processor pricing	\$400	x	2 processors per server; 3 servers	x	2 (1st year of 3 included in license)	=	\$4,800.00	\$1,600.00
Developer Tool	WebSphere Studio Site Developer v. 5.1/ per user pricing	\$1,000	x	2 seats	x	n/a	=	\$2,000.00	\$666.67
Annual Maintenance Agreement	WebSphere Studio Site Developer v. 5.1/ per user pricing	\$200	x	2 seats	x	2 (1st year of 3 included in license)	=	\$800.00	\$266.67
<b>Total</b>								<b>\$33,091.00</b>	<b>\$11,030.34</b>

**Assumptions:**

- IBM pricing based on Passport Advantage express pricing
- IBM base middleware pricing includes 1 year of support and software maintenance
- The IBM solution provides 24x7 telephone support for Severity 1 problems. Anyone on the IT staff can call-in.
- IBM server configuration is based on three 2 way servers; four servers are not required
- RedHat pricing is based on a annual subscription that includes software license, maintenance and support

### Microsoft Configuration with a Secure Customer Environment

Item	Product	Price/Unit		Units		Years of Subscription		3 Year Total Pricing	Annual Cost
Base Server OS	Windows Server 2003 / per server pricing	\$2,384.00	x	4 servers	x	n/a	=	\$9,536.00	\$3,178.67
Software Assurance (SA)	Windows Server 2003 SA/ annual pricing	\$596.00	x	4 servers	x	3	=	\$7,152.00	\$2,384.00
Client Access License	Windows CAL/ per user pricing	\$29.00	x	40 users	x	n/a	=	\$1,160.00	\$386.67
Client Access License Software Assurance	Windows CAL SA/ annual per user pricing	\$7.25	x	40 users	x	3	=	\$870.00	\$290.00
Developer Tool	MSDN Enterprise Subscription/ per user pricing	\$1,992.00	x	2 seats	x	3	=	\$11,952.00	\$3,984.00
Developer Tool Software Assurance	MSDN Enterprise Subscription SA/ annual per user pricing	\$498.00	x	2 seats	x	3	=	\$2,988.00	\$996.00
Media	Media Kit	\$23.00	x	1 kit	x	1	=	\$23.00	\$7.67
<b>Total Cost of Solution</b>								<b>\$33,681.00</b>	<b>\$11,227.00</b>

**Assumptions:**

- Microsoft pricing data is based on the Middleware Group pricing provided in Appendix18 of the study.
- Microsoft pricing is based on Microsoft Open Value. The agreement has a 3 year term period.
- The MS pricing only covers customer support during normal business hours and web support 24 x7. Only one authorized IT caller is allowed.

The tables included here by IBM are very nice, but beside the point. The specification from TMC called for 4-way servers, along with load balancing and failover. These are requirements that the lower-cost WAS Express cannot satisfy. Hence the higher-cost WAS ND is required. Also, the specification did not call for authentication against Active Directory or other LDAP source. Hence, there is no need for CALs in the Microsoft configuration. In that given configuration, the IBM software licenses are approximately 10x the price of the Microsoft software licenses. The comparison shown in the report published by TMC stands, as is.

Let us consider a hypothetical alternative specification, as proposed by IBM, where:

- 2-way server machines, not 4-way machines, are employed
- load balancing is not required (This is very impractical. Why would any customer decline to insure reliability in an enterprise application? But we shall do this for the purposes of argument)
- authentication against an LDAP source is required.

In this case, the IBM configuration would require the addition of a Tivoli Directory Server, at a cost of \$21,400 (or \$10,700 per CPU). Since Windows Server includes Active Directory, at no extra charge, there is no need to purchase an additional server license in the Microsoft side for the directory server. However, in this case, the Microsoft configuration would require CALs for device access, at a cost of about \$1100. Netting it out, in this hypothetical configuration proposed by IBM, there will still be a 3x price differential between the IBM configuration and the Microsoft configuration. And when this hypothetical business wants to include failover for reliability, they must jump to the WAS ND license, which means returning to the 10x cost differential. These are the facts. They are easily verified by asking for a real comparison quotes from IBM and Microsoft resellers.

#### **Final Microsoft note:**

In summary, the criticisms IBM raises in this document about the case study report recently published by TMC are by and large, not valid.

- The cost comparison as published by TMC stands as valid. The IBM WebSphere configuration was approximately 10x as expensive as the Microsoft configuration. In other configurations, the cost differential will vary. IBM's suggested alternatives do not fit the specification, and are incomplete in themselves.
- The suggestions that the use of ServiceDomain and Application Blocks by the Microsoft team indicate a higher level of skill than the WebSphere team, are unfounded. These interfaces and Application Blocks are mainstream, well understood, and easy to use.
- That the J2EE development team had less experience with the RRD development tool was documented in the report. However, these people were experienced J2EE developers. Their experience on this project, confronting decisions on how to structure a data access layer, how to build in concurrency, how to construct a user interface - should be considered typical of most J2EE developers.
- The productivity advantage of the Microsoft platform is real and well documented.

In conclusion, this was an exhaustive and very fair study. Is it perfect? There is no such thing. But there was no vendor bias, and full disclosure rules were again followed which is why IBM has the chance to respond in this way. That's only fair. Customers should carefully read the report, download the code and conduct their own internal analysis of WebSphere vs. .NET. The cost savings, we believe are significant with .NET. Just the software license costs alone are huge (when properly compared); but the productivity advantage is just as notable and frankly the main reason many enterprise projects are switching to .NET from J2EE. Forrester reports that 56% of North American companies are choosing .NET vs. J2EE just two years after its release. A Gartner study shows similar trends. For complete details, see:

<http://msdn.microsoft.com/vstudio/java/compare/>

Finally, customers should feel free to call The Middleware Company directly to discuss this report.

# APPENDIX

## Apology and Retraction Letter from The Middleware Company regarding the J2EE vs. .NET Pet Store benchmark that was completed in October 2002.

### A message to the J2EE and .NET Communities

November 8th, 2002

The feedback we have received about the J2EE verses .NET benchmark we posted was overwhelming to say the least. We would like to thank those who have spent the time to review our benchmark, and have provided extensive feedback on the benchmark. In response to the feedback we would like to make a few public statements about the benchmark:

- Our original intention for this benchmark was to put forth a more realistic comparison of J2EE and .NET to replace the original Pet Store comparison that Microsoft published. The TMC team worked to make extensive optimizations to Sun's implementation, which resulted in a 17x performance increase. Further information about the technical decisions we made in this project are available in our [report](#) as well as our [FAQ](#).
- However, since producing the report, we received valid feedback about additional J2EE optimizations that could be performed. We also received suggestions about using alternative J2EE architectures than the architecture we chose to test (for example, using CMP instead of BMP, or using no EJB at all).
- We have reviewed this feedback and have concluded that testing these suggestions are important and necessary to be fair to J2EE. The alternative architectures in particular may show materially different performance results. We have learned a very costly lesson here -- that even though a great deal of time, effort, and care went into this benchmark, it is important to give a complete view of how J2EE and .NET compare for the J2EE community to accept the results.
- We also admit to having made an error in process judgment by inviting Microsoft to participate in this benchmark, but not inviting the J2EE vendors to participate in this benchmark. We now realize that following this procedure is critical for the J2EE community to accept benchmark results and perceive them as credible.
- Because of the above issues, definitive conclusions about how J2EE verses .NET compares cannot be drawn unless a 2nd test is conducted which takes the above into account.

We do not currently have a public position yet about whether a round 2 will be conducted, especially given the public beating we just took. We have heard everything from accusing us of having sold our opinion to Microsoft, to that our parent company, Precise Software, is a strategic partner of Microsoft and that somehow tainted the results.

The above accusations are far from the truth, and we are disappointed that this has turned so ugly. We have done a lot of hard work for the J2EE community over the past few years, building TheServerSide.com community, writing the leading books on the technology, and much more. In fact, the reason we started this company was to help customers succeed. That is what gets us excited about our work. We tried to make J2EE perform well to show as realistic a comparison as possible, because we knew that one day we would need to post the results. The irony is that the perception right now is completely the opposite -- which this was a rush job, and that we sold our opinion to Microsoft as a marketing gimmick. This really hurts.

Going forward, we have an important decision to make. Despite our wounds, as engineers, a fair and complete performance comparison of J2EE verses .NET remains extremely interesting to us. A round 2 held in a fair and equitable fashion would be useful to get the facts out there about how J2EE and .NET compare, so people can make informed decisions. It would also be a chance to see how different J2EE architectures compare, such as BMP verses CMP, and EJB verses no EJB. We are willing to completely recode Pet Store with modern design patterns if necessary to achieve this, so that it is more benchmark-worthy -- something we did not have time to do in round 1.

A 2nd round test would fully involve vendors from both the Microsoft side and the J2EE side. It would also involve you -- the J2EE community at large. The process would be public, open, and auditable. The community would have the chance to review code as it is developed, and provide feedback to ensure J2EE is well represented. We would use many of the common performance and architectural suggestions received based on the first study. We have not made a final decision about whether to hold a round 2 benchmark, since this is a very controversial subject, with many faces. We are interested in hearing your opinion on the matter, to help us make the best decision for the community. The two questions we have for you are:

- A) Do you think we should conduct a 2nd benchmark?
- B) How do you think that benchmark should differ from the current benchmark?

We will review your feedback, take it into serious consideration, and then post our decision.

Please post your feedback here:

[http://www.theserverside.com/home/thread.jsp?thread\\_id=16398](http://www.theserverside.com/home/thread.jsp?thread_id=16398).