



# .NET 2.0: Responsive IT Solution for Your Enterprise Applications



## Solutions | J2EE to .NET 2.0 Migration

Published: March 2005

Author: Ajay Deewan

***Adapting to change has always been important to business success. In today's business environment, staying ahead means continuously adapting to change—and making change work in your favor. To do that, business leaders need IT systems that support and enable their strategic decisions. With .NET 2.0, this goal seems more achievable than ever before. Find out why you should migrate your applications to Microsoft .NET 2.0 framework and what benefits you derive***

### White Paper Overview

#### **Abstract**

This white paper talks about the comparison between the two dominating contemporary technologies, .NET and J2EE and lists down the advantages of .NET over J2EE. This also explains the Calance approach towards the J2EE to .NET migration solution.

#### **Business Situation**

Making design decision on choosing the right technology and platform for the enterprise applications.

#### **Benefits**

- Reduced cost of ownership
- Improved customer service by faster time to reach market
- High developer productivity
- Fast, secure & dependable network communications
- Enables new business opportunities and integration of systems regardless of the platform they are running on.
- Enables IT organizations to extend existing applications to integrate with business partners.
- Improved application performance, security, and easier deployments.
- Opening the door to new business opportunities by making it easy to connect with partners.

#### **Software and Services**

- Microsoft Windows® 2003 Advanced Server operating system with Internet Information Services
- Microsoft SQL Server™ 2005
- Microsoft .NET Framework
  - ASP.NET 2.0

### **Table of contents**

Table of contents.....	1
Introduction .....	2
What is .NET .....	2
What is J2EE .....	3
Why Migrate.....	4
Additional Features offered by .NET 2.0.....	7
Prior considerations for migration to .NET .....	9
Calance capabilities.....	12
Calance Approach to Migration .....	12
Glossary .....	13



## **Introduction**

Since almost two years, there has been an active debate over the endurance of the two giant enterprise platforms, Microsoft's .NET and Sun's Java 2 Platform, Enterprise Edition (J2EE). Which technology would emerge as the leading platform for developing new web applications?

Now that the combat has settled down a bit, each platform has managed to capture a substantial share of market for itself. In fact, many organizations have arranged for a strategy for one or the other of these frameworks. But many (e.g. those which have large investments in the legacy systems) are still evaluating the merits of the two platforms. The IT personnel in those businesses is questioning as to what extent would choosing .NET or J2EE influence their host strategy, etc.

It's getting harder, not easier, to pick a clear winner, because J2EE and .NET are so similar. With J2EE and .NET, selection may be based less upon intrinsic merits of the platforms and more on your existing environment (e.g. resources, investments) and personal preference or style. Instead of choosing your platform based on marketing hype or technical bias, you can look at more bottom-line factors:

- What assets does your company already possess (software, hardware, middleware)?
- What level of experience do you have in people who know both your business and your implemented technology?
- Will upgrading that system asset result in a positive ROI for the business?

Adopting this philosophy of architecture will provide the greatest stability and ROI over time, not whether you choose J2EE or .NET. Careful consideration must still be given to which technologies are chosen to support and deploy, but this philosophy provides options and time to decide which application assets can benefit from advances and which ones are just fine the way they are.

This paper assists in the decision making process of migrating from J2EE to .NET by providing the advantages of .NET over J2EE.

## ***What is .NET***

Microsoft® .NET (pronounced "dot net") is both a vision for how software should be written, and a set of tools for developing software that realizes this vision. This technology is a both a methodology and transport layer for passing information between components on different machines, different networks, and different operating systems. Integrated across the Microsoft platform, .NET technology provides the ability to quickly build, deploy, manage, and use connected, security-enhanced solutions with Web services. .NET-connected solutions enable businesses to integrate their systems more rapidly and in a more agile manner and help them realize the promise of information anytime, anywhere, on any device.

**.NET:** *.NET-connected solutions enable businesses to integrate their systems more rapidly and in a more agile manner and help them realize the promise of information anytime, anywhere, on any device.*

Language interoperability is the driving force behind .NET. It uses Common Language Runtime (CLR) software, designed for the Windows environment, which



allows a wide range of programming languages to freely share and extend each other's components. Because .NET supports more than 20 different languages, it is well-suited for organizations that have many COBOL, C++, Visual Basic®, or even Java programmers.

### **What is J2EE**

J2EE is a platform from Sun Microsystems based on a series of standards for developing Java applications and implementing web services. It has an enterprise-level component strategy with its Enterprise JavaBeans (EJB) architecture that is geared to future scalability. EJBs also allow for multiple user interfaces. In addition, J2EE has servlets (Java applications running on a web server or application server), and JavaServer Pages (JSPs), which simplify the process for presenting dynamic content on web pages.

The cornerstone of J2EE is the Java object-oriented programming language, which allows developers to write software for one platform and execute it on another, with no modifications. Portability is a key feature of the J2EE framework.

***J2EE:** J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applications.*

## **Why Migrate**

Although the rivalry between .NET and J2EE continues, neither platform is expected to dominate business-application development in the near term. Instead, their roughly equal capabilities will win roughly equal market share for .NET and J2EE. That means the two technologies will be used in 80 to 90 percent of business-application development over the next five years.

IT organizations can choose one or both of these de facto standards (or even defer the decision for a while). Ordinarily, only large businesses are in a position to adopt both .NET and J2EE, due to two circumstances: 1) they have sufficient resources to train their development staff on both platforms, and 2) they have the capacity to develop best practices for managing environments that include elements from both platforms.

Unlike their larger counterparts, small and midsize organizations won't have the luxury of supporting both platforms simultaneously. Due to limited resources, they will probably be forced to choose between .NET and J2EE. And because Microsoft has established a strong presence in small and midsize businesses, .NET can reasonably be expected to prevail in this market.

The following table gives a full listing of equivalences between J2EE and the .NET platform:

<b>Technology</b>	<b>.NET</b>	<b>J2EE</b>
<b>Support technologies</b>		
Distribution protocol	DCOM, SOAP	RMI/IIOP
Firewall	ISA	not specified
HTML page caching	ISA, ASP.NET	not specified
<b>Presentation tier technologies</b>		
Infrastructure	IIS	not specified
Programming model	ASP.NET	Servlets, JSP
High availability	NLBS, ACS, other	not specified
Load balancing	NLBS, ACS, other	not specified
Management	ACS	not specified
<b>Middle tier technologies</b>		
Infrastructure	COM+	EJB
Programming tool	Visual Studio.NET	not specified
High availability	ACS	not specified
Load balancing	ACS	not specified
Security API	COM+ Security Call Context	JAAS
Message Queue API	MSMQ	JMS 1.0
Asynchronous components	Queued (COM+)	Message driven beans (EJB 2.0)
Naming and Directory Service	ADSI	JNDI

*How do I map technical jargon of .NET and J2EE?*

<b>Data tier technologies</b>		
Distributed transaction	MS-DTC	JTS
Relational DB API	ADO.NET	JDBC 2.0
Hierarchical DB API	ADO.NET	-
Database storage	SQLServer	-
Mainframe DB connectivity	HIS	Java Connectors
<b>Framework technologies</b>		
eCommerce framework	Commerce Server	-
Business to business orchestration	BizTalk Server	-

As is apparent from the above table, many of the .NET platform functional areas have no counterpart in J2EE, although the above listing points out the similarities between the two technologies.

### Advantages of .NET 2.0 over Java

The .NET platform offers a number of advantages that make it particularly appealing if you now have applications developed in Java. A few of the generic key benefits include:

- ❖ **Framework Classes.** Because Java is multiplatform, the set of framework classes has traditionally been limited to what is available on all platforms. This affects both the scope and richness of classes: The scope is limited to the lowest-common denominator of available computers and operating systems. This also affects the richness of the classes themselves. The built-in Java classes are typically very simple, and developers often build their own framework on top of them, or have to find third-party frameworks — for example, XML document parsing. In order to parse XML documents in Java before the release of JDK 1.4, you have to either use a third-party library, or write your own parser. Writing your own parser commonly represents more than 100 lines of code.

The .NET Framework is built upon the Windows platform. A version of the common language runtime and .NET Framework class library is also available for FreeBSD. The classes offer the full scope of what is available on this platform. Additionally, the classes are richer than what is available in Java: .NET provides a programming framework rather than a base on which to build a programming framework.

- ❖ **Write Once, Debug Everywhere.** While the Java ideal of "write once, run anywhere" is attractive, most applications written today target only one operating system. There are two technical reasons for this: Because the Java framework is limited in scope and richness, developers often use proprietary classes to access features available to the target platform. Secondly, because lingering incompatibilities between the various Java implementations have

*Why should I migrate my existing enterprise applications to .NET framework (2.0)?*

continued to plague true cross-platform efforts, developers must test their code on every platform they wish to support. Because of this, some developers have half-jokingly referred to Java as the "Write once, debug everywhere" platform.

- ❖ **Security.** Security is richer in .NET. Along with APIs for cryptography, secure cookies, and authentication, there is also a rich and extensible mechanism for assigning different levels of permissions to different sets of users, based on where the code came from. For example, all code from Microsoft can be treated as safe. All code from a specific URL can be granted (or denied) read/write access to specific resources such as the temp directory or screen. In addition, applications can be marked as requiring certain permissions (for example, requires access to the file system). If these permissions are not available, then the application will not load. Because the security check is performed at load time, the developer is freed from writing security fail detection logic throughout the application. This is managed through a new technology called "code access security," which works for code loaded from either the Internet or the local machine.
- ❖ **Increased performance.** Where .NET outshines Java is in the ability to fine-tune application behavior and performance. The core application building blocks are designed for scalability and high throughput. For example, with ADO.NET disconnected data access, more data functions can be completed in-memory, which in Java Database Connectivity (JDBC) require a round trip to the database. In addition, the .NET application architecture is designed to be faster. For example, ASP.NET Web pages provide a huge performance improvement over Active Server Pages (ASP), Java Server Page (JSP), and HTML embedded script languages. When compared to the Java platform, .NET performance is not only better in single-user scenarios, but also in massively multi-user scenarios.
- ❖ **Increased productivity:** According to a study performed by Software Productivity Research (SPR), Visual Studio .NET is capable of increasing development productivity between 35% and 55%.
- ❖ **Less source code:** J2EE requires more source code than .NET to obtain the same functionality. Based on the Software Productivity Research data, coding a function point in Java requires about 53 statements while .NET requires only 16. In fact, a .NET implementation of the Nile application was written with only 3,484 lines while the Java version needed 14,273 lines
- ❖ **More scalability:** Scalability refers to the ability to handle more workload, typically from the addition of more users. A study developed by ObjectWatch, Inc. states that the .NET platform has a major advantage in total cost of ownership vs. J2EE: You can expect to spend five to ten times as much to handle the same workload using a J2EE/Unix platform as would be required with the .NET/Windows platform.

*What do I gain from the migration?*

- ❖ **.NET is a multi-language development platform:** .NET is not merely a language, but a complete platform with a common architecture that facilitates developing, debugging and testing multi-language applications.
- ❖ .NET has Microsoft's A-team marketing it
- ❖ .NET released their web services story before J2EE did, and thus has some mind-share
- ❖ .NET has an awesome tool story with Visual Studio.NET
- ❖ .NET has a simpler programming model, enabling rank-and-file developers to be productive without shooting themselves in the foot
- ❖ .NET gives you language neutrality when developing new eBusiness applications, whereas J2EE makes you treat other languages as separate applications
- ❖ .NET benefits from being strongly interweaved with the underlying operating system

### ***Additional Features offered by .NET 2.0***

- ❖ **Reduce the number of lines of code required by 70%.** The declarative programming model freed developers from having to write reams of code, but there are still many scenarios where this cannot be avoided.
- ❖ **Localization.** As more and more companies reach out to other countries for business, creating global Web applications with Microsoft ASP.NET is becoming more and more important. ASP.NET 2.0 makes it even easier to provide support for multiple cultures and locales through improved runtime and tool support.
- ❖ **Personalization.** .NET 2.0 greatly simplifies and enhances personalization by providing integrated controls and functions, which are designed specifically to support each phase of the personalization process. From an ASP.NET 2.0 perspective, the three main areas of personalization support are membership, profiles, and Web Parts, providing a rich personalized experience. Development of portals like <http://my.msn.com> has become easier and more maintainable with .NET 2.0.
- ❖ **Interoperability, Integerability & Extensibility.** Web services are revolutionizing how applications talk to other applications—or, more broadly, how computers talk to other computers—by providing a universal data format that lets data be easily adapted or transformed. Based on XML, the universal language of Internet data exchange, Web services can communicate across platforms and operating systems, regardless of the programming language in which the applications are written.

**Localization:**  
*Implementation of localization into your applications has become a necessity rather than an added feature. .NET 2.0, developing internationalized applications is easier than ever before.*

**Personalization:**  
*Development of portals like <http://my.msn.com> has become easier and more maintainable with .NET 2.0.*

- ❖ **Portability: Use a single control set for all devices.** Mobile devices are becoming more pervasive, with an increasing number of new devices. Many of the server controls render appropriately for small screens, but there are two major problems with the current support for mobile devices:
  - Having a separate set of server controls purely for mobile devices is not only confusing but also costly, and
  - Adding support for new devices requires additional development work and maintenance. ASP.NET 2.0 will provide a single set of controls and an extensible architecture to allow them (and other controls) to support multiple devices.
- ❖ **Provide the fastest Web server platform.** Although ASP.NET 1.0 offers a fast server platform, ASP.NET 2.0 will improve areas such as application start-up times and provide better application tracing and performance data. Innovative caching features will enhance application performance, especially when SQL Server is used.
- ❖ **Provide easier and more sophisticated management features.** Administration of ASP.NET applications under version 1.x required manual editing of the XML configuration file, which is not a great solution for administrators. Version 2.0 brings a graphical user interface-based administration tool that is integrated with the Internet Information Services (IIS) administration tool.
- ❖ **Easy implementation of entire scenarios.** The better management features are built on top of a management application programming interface (API), allowing custom administration programs to be created. Along with application packaging this will provide support for easily deployable applications, with or without source.

One of J2EE's major disadvantages is that the choice of the platform dictates the use of a single programming language, and a programming language that is not well suited for most businesses.

With all of the above points under consideration, .NET clearly emerges as the winner. Summarizing all of the above, you will need to consider the skills of your current workforce, along with your interoperability, security, and performance needs. The competition between .NET and J2EE is fierce, and therefore they will each continue to push the other to improve. So, use of a little common sense in the decision making process will make it hard to make the wrong decision. The following table further helps in the process.

*How do I decide which framework to choose?*

Decision Criteria	J2EE	.NET
Your data is stored in Oracle/DB2	X	X
Your data is stored in SQLServer		X
Your team programs in Java	X	X

Your team programs in many languages		X
Vendor neutrality is important to you	X	
Platform integration is important to you		X

**Prior considerations for migration to .NET**

With the advent of Microsoft’s .NET framework and Visual Studio, you cannot afford to be a bystander. You need to be a part of the game if you are in any way linked to the Microsoft and Windows. You should consider the following points before moving your existing applications to .NET:

- ❖ **Evaluate Project Status.** If you live in a Microsoft world, you need to begin embracing the .NET lifestyle—but that doesn't mean you must migrate all your projects at once. Rather, evaluate each project to determine whether it's an immediate candidate for .NET migration, if you should migrate it later, or not at all. Start with the project timelines and determine where each of your applications is in the development life cycle. For new projects, consider developing with the .NET Framework. Projects already in development, on the other hand, shouldn't be converted to .NET until the next major release because changing platforms midstream can be costly and time consuming. You should migrate existing, completed applications or components that need to interface with your new .NET applications and components first. Because you incur a performance penalty for calling unmanaged COM components from .NET managed code components (and vice versa), you'll want to eliminate that bottleneck as soon as possible. If you have applications and components that exist in a vacuum, and are adequately meeting business needs, you don't need to convert them to .NET in the immediate future.
- ❖ **Determine Your OS:** The operating system (OS) platform you plan to target can also have a substantial impact on your .NET migration plan. When paired with Windows 2000 or Windows XP, you can take full advantage of the .NET Framework's feature set. Therefore, if your user base hasn't upgraded to one of the latest Windows operating systems, you might want to explore upgrading them at the same time you do your .NET migration. Although the .NET Framework works on Windows 98 and Windows ME, these platforms don't offer full access to everything .NET offers. For example, .NET applications deployed on Windows 98 or ME won't have local COM+ support, because it's not baked into the OS (Note: Microsoft recently renamed COM+ as .NET Enterprise Services). Also, on these two OSs, the Internet capabilities of the .NET Framework will be limited. Of course, if you're

*Any caveats in migrating to .NET 2.0?*

moving to an ASP.NET Web application architecture, the client OS is less of a factor (possibly none at all).

- ❖ **Prepare Your Staff.** Upgrading to .NET is a significant transition, and your network and server support staff needs to be prepared. Don't assume that just because the developer community has high interest in .NET that your deployment team has adequate core competencies to take your .NET applications to market. Get your server administrators trained in Windows .NET Server, and working with your development teams on the numerous XML-formatted configuration files associated with ASP.NET Web applications and Web services. Also, have your server administrators explore the advantages of the other Enterprise .NET Server products such as Internet Security and Acceleration (ISA) Server 2000, Application Center 2000, and SQL Server 2000. All these tools are built from the ground up to interact with .NET applications.
- ❖ **Decide on Delivery:** A large factor to consider when migrating your applications to .NET is the medium by which you want to deliver the solution. ASP.NET has matured into a feature-rich, event-driven Web development platform, so consider turning some of your old client/server applications into Web-enabled applications. Because .NET doesn't provide native support for Distributed Component Object Model (DCOM), you should re-architect these components into Web services. VS.NET simplifies this process: You can transform a business logic component into a Web service by adding a few class attributes, and recompiling your project. All the Web services plumbing is done behind the scenes, which allows you to concentrate on porting your business logic. You'll likely find the Web services approach eases many of your deployment and configuration woes because it can operate through network firewalls and across platform boundaries. If you build a Web-enabled application, you'll also reduce the time it takes to create interfaces to your application for wireless devices, such as mobile phones and personal digital assistants (PDA).
- ❖ **Choose a Language:** You'll have to make several design standards decisions when migrating your existing applications to .NET or developing new .NET applications. Start with the language your developers will use for your development. The upgrade wizard that ships with VS.NET doesn't do a complete job and requires substantial work to make your application truly .NET compatible.

If you also have C++ or Java developers on staff, you might want to consider Microsoft's new development language, C#, as your language of choice. Of course, all the various .NET languages are interchangeable, but you'll want to settle on one or two languages to ensure you can maintain your code base.



Cool!!!

- ❖ **Take Advantage of OO:** Another design consideration to weigh is how you'll take advantage of the new object-oriented (OO) features of .NET. You'll definitely want to reassess the design structure of the objects in your system, and their relationships to each other. Do this *before* you port your legacy code, because these OO features might allow you to eliminate large sections of code.
- ❖ **Select a Naming Convention:** The naming conventions for developing with VS.NET have changed from previous versions of Visual Studio; Microsoft no longer recommends Hungarian notation (using data type specific prefixes on variable names). Instead, it suggests camel casing for variables (the first word of the name is lower case, followed by words beginning with upper case letters) and Pascal casing for class, property, method, and event names (each word of the name begins with a capital letter). Your developers should also arrange classes in vertical namespace hierarchies such as:

System.Web.Caching.Cache

The addition of function overloading to VB.NET eliminates the need to create several methods whose names differ only slightly based on the parameter lists passed into them. In .NET, developers can roll all these methods into a single method name with different parameter lists. There's no official rule that states developers must follow these naming conventions, but doing so allows code to fit in better with the rest of the .NET Framework.

- ❖ **Consider Object Deployment:** There are some interesting deployment factors to consider when moving to .NET. The .NET Framework allows developers to decorate class declarations with attributes that indicate the exact .NET Enterprise Services behaviors necessary for them to function properly. This places object configuration in the hands of the people who are most knowledgeable on the subject.
- ❖ **Start Communicating:** You have several object communication options to choose from when you move to .NET development. I've already suggested replacing DCOM with Web services, but you might also choose to implement .NET Remoting (see Resources). Remoting provides both a performance optimized proprietary data format for making procedure calls across the network, similar to DCOM, and an XML serialized data format for interoperability across platforms. Remoting's advantage over DCOM is it's configured completely through human readable configuration files. Therefore, you can tweak the configuration settings using a simple text editor, rather than rely on specific tools that might or might not exist. If you have ever attempted to configure the Internet Information Services (IIS) metabase, you'll appreciate this feature immensely. The .NET Framework is also tightly integrated with Microsoft Message Queue, which offers a solution for guaranteed asynchronous message delivery.

*Hmm, I think I am in a better position to decide whether to move a particular application to .NET 2.0 or not.*

- ❖ **Explore Your XML Options:** Part of your .NET migration plan should also include exploring the various XML options because the .NET Framework BCL includes an extensive set of classes for XML data manipulation. For example, you might want to leverage SQL Server's ability to emit resultsets as XML. Using the .NET XML classes, you could translate (possibly through Extensible Stylesheet Language Transformations [XSLT]) and display results from a database query without ever processing the data in tabular format. Or, you could use the .NET XML classes to process an ADO.NET data set class serialized into XML form. Consider all these issues when you determine which .NET features to implement.

*Get Set Go!*

Although there's no way to cover all the factors you might encounter as you each migrate to the .NET Framework, I hope that the issues, considerations, and details presented here will help as you evaluate what your organization needs to get started on the platform expected to drive Microsoft applications for the next decade.

## ***Calance capabilities***

- ❖ **Advanced Technology Group.** Calance has constituted an Advanced Technology Group for .NET 2.0, which is focused on doing an extensive research & development of cutting edge business application architectures in the .NET 2.0 framework. The team also comprises of expert J2EE architects to develop custom migration solutions for customers who are planning to migrate from J2EE to .NET 2.0.
- ❖ **Rational Unified Process (RUP).** Calance after pioneering RUP approach to software development in various technologies like J2EE, .NET 1.x , has come up with a proprietary application development process especially to manage .NET 2.0 projects. This translates into providing our customers, the same level of delivery commitments in their .NET 2.0 projects.

*Why Calance?*

## ***Calance Approach to Migration***

Calance employs a 2 phased approach to implement migration solutions:

- ❖ **Phase 1 - Convert for Functional Equivalence.** The primary goal for a Phase 1 conversion is to create a Web site in C# and ASP.NET that is functionally equivalent to the existing JSP/Servlet/Java Web site. In order to make this conversion possible, the following two primary tasks are performed:
  - **Code Analysis** - Identifying critical functionality and potential trouble spots provides the basis for the conversion.



- **Convert with the JLCA and establish functional equivalence** - The Java Language Conversion Assistant converts as much of the site as possible, and produce a comprehensive report of problem areas. The JLCA successfully converts as much as 90% of the website's code, however, we still need to act on any errors reported by the JLCA. The goal is to establish functional equivalence with your original Web site.

*Calance migration projects methodology revealed!*

❖ **Phase 2 - Optimizing Site For ASP.NET.** Once we have a working ASP.NET Web site, we start leveraging ASP.NET features and optimizing the code for ASP.NET and IIS. In this part of the conversion process, we replace converted code with more appropriate ASP.NET code, introduce new features to take advantage of ASP.NET and IIS, and re-architect the site for improved security, reliability, and performance. This phase is divided into two primary steps:

- **Leveraging ASP.NET Features** - The first step of Phase 2 is to start leveraging simple ASP.NET features to improve performance, increase security, reduce the code, and simplify the projects. In this step, we replace tag libraries and Javascript with ASP.NET Web controls, leverage the CodeBehind file, and perform other simple changes that are only possible once we have broken the connection with JSP.
- **Best Practices** - The last step in a conversion project is to evaluate the application and re-architect to take advantage of ASP.NET best practices. In this section, we leverage material from Microsoft's Perspective Architecture Group to evaluate the site security, improve performance, and increase the scalability and stability.

## Glossary

- ADO.NET - The .NET API for accessing the data tier.
- Application Center Server - The .NET technology that provides middle tier load balancing and general management of cluster farms.
- Assembly - The unit of application deployment for the .NET platform.
- BizTalk Server - The .NET technology that provides orchestration between business processes, especially legacy business processes.
- C# - A new C++ derivative programming language that is similar in functionality, look, and feel to Java.
- CICS - An IBM mainframe transactional processing monitor technology.
- COBOL - One of the primary business programming languages.
- COM+ - The .NET middle tier infrastructure designed to support business components.
- Commerce Server - The .NET eCommerce framework.
- CORBA (Common Object Request Broker Architecture) - The earliest of the component architectures defined by the OMG.
- Common Language Runtime - The .NET runtime that provides support for compiled IL.
- DB2 - An IBM relational database product that competes with SQLServer.
- DCOM - The Microsoft protocol typically (but not exclusively) used for communicating to components within the .NET architecture.

- eBusiness - A company that does all or an important part of its business over the Internet.
- eCollaboration - Business to business collaboration conducted over the Internet
- eCommerce - Business that is conducted over the Internet.
- Eiffel - An important research programming language, used in some businesses.
- EJB (Enterprise JavaBeans) - The J2EE middle tier infrastructure designed for business components.
- Entity Beans - A component model that is specific to the EJB technology that provides varying degrees of state management.
- HIS (Host Integration Server) - The .NET technology that interoperates with IBM mainframes.
- HTML - The industry standard for describing browser displays.
- HTTP - The industry standard for communications over the Internet.
- IL (Intermediary Language) - The intermediary language used by the .NET platform.
- IMS - An IBM mainframe database technology.
- ISAS (Internet Security and Acceleration Server) - The .NET technology that provides firewall protection and HTML page caching.
- ISV (Independent Software Vender).
- J2EE - Java 2 Enterprise Edition.
- JAAS (Java Authentication and Authorization Service) - The J2EE API for managing security.
- Java - The programming language associated with J2EE.
- Java Applets - A packaging technology for downloading and running Java code on the client side in a browser.
- Java 2 Enterprise Edition - A general term describing Sun's family of eCommerce specification.
- Java Byte Code - The intermediary language run within the Java Virtual Machine.
- Java Connectors - The J2EE API used for plugging in different database technologies (not programmer visible).
- Java Server Pages - The J2EE technology that, along with Java Servlets, is the programming model for the presentation tier.
- Java Servlets - The J2EE technology that, along with Java Server Pages, is the programming model for the presentation tier.
- Java Virtual Machine - The Java language environment.
- JDBC (Java Database Connection) - The J2EE API for accessing the data tier.
- JMS (Java Message Service) - The J2EE API for accessing message queues.
- JNDI (Java Naming and Directory Interface) - The API for naming and locating specific instances used in J2EE.
- JTS (Java Transaction Server) - The J2EE API used for managing transactions boundaries, largely superseded by EJB automatic transaction boundary management.
- MS-DTC (Microsoft Distributed Transaction Coordinator) - The .NET technology that manages the two phase commit protocol used in distribution transaction coordination.
- MTS (Microsoft Transaction Server) - The original middle tier, component based infrastructure designed to support highly scalable applications.
- .NET - A general term for describing Microsoft's eCommerce platform.
- .NET Building Block Services - Collaborative services that help businesses work together.
- .NET Client Systems - Subsets of the .NET platform designed to support various thin client platforms.

- .NET Enterprise Servers - A collection of optional enterprise level products available for the .NET platform.
- .NET Framework - The overall infrastructure in which .NET applications run.
- NLBS (Network Load Balancing Service) - A .NET technology for load balancing and providing high scalability to the presentation tier.
- OMG (Object Management Group) - An early industry consortium that defined the original CORBA specifications.
- RMI/IIOP (Remote Method Invocation over Internet InterOrb Protocol) - The protocol used for communicating with components in J2EE based on the CORBA protocol.
- SOAP (Simple Object Access Protocol) - An industry standard for packaging of method requests.
- SQLServer - Microsoft's high-end database product.
- TCP/IP - A low-level communications protocol powering the Internet.
- TPC-C - An industry standard benchmark based on a distributed order entry system for measuring throughput at the middle tier/data tier combination.
- Visual Studio.NET - The programming environment for .NET.
- WebLogic - BEA's eCommerce platform, a superset of J2EE.
- WebSphere - IBM's eCommerce platform, a superset of J2EE.
- web service - A middle tier business component that is callable over SOAP.
- URL (Universal Resource Locator) - A standard Internet location.
- XML - An industry standard for architecting text strings.

---

#### **About the author**

Ajay Deewan is currently working as an Architect in Calance's Advanced Technology Group for Microsoft Technologies. He has wide-ranging experience in enterprise solutions. He has worked in the areas of software project development and solution implementations esp. on Distributed Technologies and e-Commerce. He has rich experience on some of the best products available in the market viz. Forte, Vignette Development Center, Informatica, MicroStrategy, Rational Rose, etc. He has worked extensively in some of the big corporations like Cognizant and Sapient, and has written success stories on more than 14 projects in a short span of last 4 years.

---

---

For more information about Calance products and services, call the Calance Sales Information Center at (412)-455-5469. To access information on internet, go to [www.calance.com](http://www.calance.com).

© 2005 Calance Corporation. All rights reserved.

This case study is for informational purposes only. CALANCE MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. Calance and the Calance logo are either registered trademarks or trademarks of Calance Corporation in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

