

Microsoft .net Framework

**A Guide to Reviewing the Microsoft .NET Framework:
a platform for rapidly building and deploying XML Web services and
applications to solve today's business challenges.**

Abstract

The Microsoft .NET Framework is a platform for building, deploying, and running XML Web services and applications. It provides a highly productive, standards-based, multi-language environment for integrating existing investments with next-generation applications and services as well as the agility to solve the challenges of deployment and operation of Internet-scale applications.

For the latest information on the .NET Framework, visit <http://msdn.microsoft.com/net>

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2001 Microsoft Corporation. All rights reserved.

Microsoft, the .NET logo, ActiveX, JScript, Visual Basic, Visual C++, Visual FoxPro, Visual InterDev, Visual J++, Visual Studio, Win32, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft 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.

*Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA
1297*

CONTENTS

REVIEW CRITERIA.....	1
Overview	4
 .NET FRAMEWORK: INTRODUCTION	4
Web Development	4
Challenges	5
Solution: XML Web Services	5
The .NET Framework Design Goals	6
 THE XML WEB SERVICES PROGRAMMING MODEL	8
Core XML Web Services Technologies SOAP Family	8
 .NET FRAMEWORK: OVERVIEW.....	10
The .NET Framework: The Microsoft XML Web Services Engine	10
THE .NET FRAMEWORK: Three Parts	10
The Common Language Runtime	10
What the Programmer Sees: The Unified Classes	13
 DEVELOPER PRODUCTIVITY	16
Use any programming language	16
Take advantage of industry-leading tools	16
Developers write less code	16
Employ Windows 2000 Applications Services	17
Create XML Web services Easily	17
 AGILITY TO SOLVE TODAY'S BUSINESS PROBLEMS	21
Deliver on Software as a Service	21
Supporting ECMA standards for C# and CLI	21
Device support for broad reach	21
Extend all existing software transparently	21
Access databases easily with Microsoft ADO.NET	21
Web Data Access	21
Benefits of ADO.NET	24
 IMPROVED OPERATIONS	26
Evidence-based security	26
Internet Capable Security	26
Simplify application deployment	27
Run more reliable applications	29
Improve performance	29
 .NET FRAMEWORK FEATURE SUMMARY	30
XML Web Services Standards Support	30
Developer Productivity	30
Agility to solve today's business challenges	31
Building Next-Generation Web Applications	35

SUMMARY	35
Loosely Coupled Designs Enable Scalability	35
Leverages Operating System Services	35
Multi-Language Support	35
Standards Based	35
APPENDIX A FREQUENTLY ASKED QUESTIONS.....	36
APPENDIX B .NET FRAMEWORK TOOLS.....	43
APPENDIX C GLOSSARY	46

REVIEW CRITERIA

The Microsoft® .NET Framework and Microsoft Visual Studio® .NET belong, in the broadest sense, in the *software development* category. The leading edge of the category, according to customers and industry analysts, is the *distributed application development* segment. The software development category contains all developer tools products, while the distributed application development segment includes only those tools aimed at server- and client-side application development. Microsoft believes the main contenders in this segment include the .NET Framework and Visual Studio .NET, versus the Java-based product offerings of IBM WebSphere and VisualAge, and Sun iPlanet Application Server and Forte.

If you examine the needs of developers and enterprise organizations, you can create a list of criteria for evaluating the .NET Framework and Visual Studio .NET that enables the rapid development, simple integration, and improved operation of the next generation of XML Web services and applications.. These include:

Improved Time to Market Criteria

- **The ability to use any programming language.** Enable developers the ability to use any programming language. Allow applications written with different programming languages to integrate deeply with each other. Leverage current development skills to go forward without retraining. Customers have discovered the exorbitant cost of finding developers skilled in some specialized technologies. Prominent industry analysts recommend using existing skills whenever possible, and avoiding the high cost and high failure rate of relying on the technical skills in shortest supply.
- **Access to state-of-the-art development tools.** Deliver well designed development tools such as integrated debugging and profiling.
- **Improved code design.** Provide a highly componentized, plumbing-free design that enables developers to focus on writing business logic. Eliminate the need to generate IDL or Registry code. Provide existing, well-tested controls that encapsulate common programmer tasks.
- **Support for loosely coupled and tightly coupled architectures.** To achieve performance, scalability, and reliability in distributed applications, there are some operations in which components can be tightly coupled, that is, interdependent in real time. Most modern systems have this capability. However, successful platforms must also have complete support for loosely coupled, message-oriented communication, in which the chain of events supporting a distributed transaction is not broken or blocked by any real time dependencies. Both architectures must be supported completely and naturally so developers create scalable solutions by default.

Simple Integration Criteria

- **Providing software as a service.** Build and deliver software as a service. Employ standards and protocols built on XML and the SOAP family of integration standards. Provide the ability for applications to easily to share and gain access to these services.
- **Approved standards for language and infrastructure.** Standards are core to delivering software as a service. Consequently, Microsoft has submitted the specifications for the C# programming language and a subset of the .NET Framework called the common language infrastructure to ECMA and ECMA is standardizing them. These specifications are a collaboration of six other ECMA partners including Hewlett-Packard Co and Intel and are currently on track to be considered by the ECMA General Assembly later this year for formal approval.
- **Transparent extensions for legacy components and applications.** Provide the ability to integrate existing software into a language-independent platform with proven technology.
- **Ease of data access.** Make available a productive interface to any database, one that is designed for Web applications particular use of loosely coupled style of data access. Support XML as its native data format.

Improved Operations Criteria

- **Providing evidence-based security.** Provide a evidence-based-security model, one that exhibits a fine-grained, method-level control over what applications can and can't do based on who wrote the code, what it's trying to do, where it was installed from, and who is trying to run it.
- **Simplifying application deployment.** Support a simplified application deployment method that makes installing applications as easy as copying them into a directory. Build in the ability of a framework to detect damaged applications, and *self-heal* to solve the problem.
- **Increased reliability for applications.** Deliver technologies that make applications more reliable such as high performance automated memory management and event-driven monitoring of Web applications with automated granular control of restarts.
- **True performance improvements.** At the end of the day, improve the performance of typical Web services and applications.
- **A comprehensive solution that provides all of the services needed to produce an enterprise-class solution.** These include application services such as an object request broker, transaction-processing monitor, complete scripting engine, feature rich Web server, world-class messaging support, monitoring, and management infrastructure.

All of these services should be integrated and designed to work together, so developers can spend their time building applications, not cobbling together application services.

- **A consistent and unified programming model for applications running within, and applications running across, Internet firewalls.** Related to loosely coupled architectures, this criterion seeks out the solutions that enable organizations to learn, build, and support a simple programming model for intranet- and Internet-based applications in order to reduce training and support costs.
- **A consistent and unified programming model for broad reach Web-based clients, as well as rich client interfaces, and emerging smart devices.** Customers today are evaluating the full range of client devices, including PC, browser, PDA, and cellular telephone. Only by having a programming model that elegantly supports all of today's user experiences, and those that will emerge in the future, and contains a simple model for maintaining consistent business logic across any client architecture, will complex, distributed systems be successful.

A platform embracing the above principles will enable customers to lead in their respective industries, to build, deploy, and maintain better applications, and to bring new software and new services to market faster and less expensively than competitors. The .NET Framework satisfies these requirements more completely and effectively than any other product available today, and will do so into the foreseeable future. We invite you to examine these new tools carefully, and make your own comparison.

Overview

The .NET Framework is the result of two projects. The goal of the first project was to improve development on Windows, looking specifically at improving COM, the Microsoft Component Object Model. The second project aimed at creating a platform for delivering software as a *service*. These two projects came together more than three years ago. The finished product dramatically improves programmer productivity, ease of deployment, and reliable application execution, and introduces a totally new concept to computing: that of Web Services – loosely coupled applications and components designed for today's heterogeneous computing landscape by communicating using standard Internet protocols such as XML and SOAP.

Web Development

When it was created, the Web was basically a read-only file system with the added benefit that it used industry standards and protocols, enabling easy access to the contents of files. The few Web sites that were interactive were typically outward extensions of existing two-tier applications.

Early Web development was typically done with the C programming language and the Common Gateway Interface (CGI), with which very few programmers had experience. As a result, development costs for dynamic Web applications were high.

In addition, most of these Web applications were built on two-tier architectures, causing challenges around scalability and application integration. Developers simply did not design Web applications to be used by anything other than the Web page they were hosting; in other words, the user interface and the application logic were the same thing. Consequently, it was difficult to link Web applications together to form more interesting aggregations. An example of this problem would be a Web site that sells curtain rods but does not offer curtains, forcing potential customers to visit at least two separate sites to purchase a complete solution for their window treatments.

As a result of advances in the Microsoft Component Object Model (COM) and the release of technologies such as Microsoft's Active Server Pages (ASP) in 1996, Web sites offered a more interactive user experience. ASP does this by making it easy to call the business logic and platform services that developers need through simple script languages. COM support makes it easy to write applications through its ability to package this business logic into modular units that can be written in a wide range of popular programming languages, such as Microsoft Visual Basic®, C++ or COBOL.

Web sites are now offering richer user experiences, and taking basic steps to overcome some of the challenges of application integration, with tricks such as using HTML frames to embed one company's Web site within another and HTML "screen scraping" to extract data from Web pages.

But these strategies for application integration have shortcomings. Simply put, they are brittle: What happens if the other Web site changes its content to promote a competitor or the company goes out of business, leaving the page with a broken link?

Advancements in Web development are rapidly moving from this two-tier architecture to a N-tier design, which enables a richer integration strategy by exposing business objects or middle-tier logic to Web and partner integration. The challenges with trying to use encapsulated business logic in this way is that most of these applications are designed on tightly coupled, proprietary protocols.

Challenges

To date, the companies that have tried to offer solutions for enabling a Web site to expose application integration information and functionality in a modular, scalable, and Internet-friendly way, have encountered significant challenges. Chief among these challenges are the following:

Time to market. The length of development time for getting an application or Web site to market may render the offering no longer viable.

Scaling to the Web. Existing object models and component designs simply do not work over Internet protocols. Stateless application development that can be rerouted and served by any server is a foreign concept for many developers. Yet such a design pattern is vitally important to achieve global scalability.

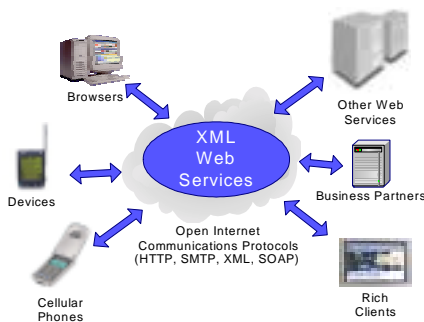
Lack of end-to-end development tools. Toolsets available today don't empower organizations with the flexibility necessary to stay ahead of their competitors. In the rapidly changing world of the Internet, organizations must exhibit the agility to integrate new partners, with development tools that solve the challenges of today's heterogeneous computing environments.

Solution: XML Web Services

To solve the challenges facing Internet development now and for the future, we need to be able to write applications in any programming language, access any platform, and scale over the Internet to global proportions. This application development strategy is very compelling, as it enables companies to make use of existing hardware, utilize current applications and use developers they have on staff, without having to retrain them on a new programming language.

This style of computing is called XML Web services and represents the next evolution of application development. An XML Web service is an application that exposes its functionality programmatically over the Internet or intranet using standard Internet protocols and standards such as HTTP and XML.

XML Web services solve the challenges facing Web developers by combining the tightly coupled, highly productive aspects of N-tier computing with the



An XML Web service is an application that exposes its functionality programmatically over the Internet or intranet using standard Internet protocols such as HTTP and XML.

loosely coupled, message-oriented concepts of the Web. Think of XML Web services as component programming over the Web.

Conceptually, developers integrate XML Web services into their applications by calling “Web APIs” just as they would call local services. The difference is that these calls can be routed across the Internet to a service residing on a remote system. For example, a service such as Microsoft Passport could enable a developer to provide authentication for an application. By programming against the Passport service, the developer can take advantage of Passport’s infrastructure and rely on Passport to maintain the database of users, make sure that it is up and running, backed up properly, and so on, thus offloading a whole set of a development and operational chores.

The .NET Framework Design Goals

The .NET Framework is the culmination of the combined efforts of several teams at Microsoft, working together to create a platform for rapidly building and deploying XML Web services and applications. The vision for the .NET Framework platform is to combine a simple-to-use programming paradigm with the scalable, open protocols of the Internet. To achieve this vision several intermediate goals had to be delivered.

Integration through Public Internet Standards

To communicate with business partners, customers, geographically separated divisions and even future applications, development solutions need to offer both support for open Internet standards and deep, transparent integration with these protocols that doesn’t force the developer to learn this underlying infrastructure.

Scalability through a Loosely Coupled Architecture

The largest, most scalable systems in the world are built on message-based asynchronous architectures. But building applications on such an architecture is complex, and the tools are often less rich than those of more tightly-coupled, N-tier application development environments. The .NET Framework was built to bring together the productivity advantages of tightly-coupled architectures with the scalability and interoperability advantages of loosely-coupled architectures.

Multi-Language Support

Developers use different languages because the languages are good at particular things: some languages are particularly adapted at mathematical manipulation; some have particularly rich financial functions, and so on. Rather than forcing everyone to learn a single programming language, the .NET Framework enables applications written in many different programming languages to integrate deeply with each other. As a result, the .NET Framework enables companies to take advantage of their current development skills without the need of retraining, and allows developers to program in the language of their choice.

Enhancing Developer Productivity

With the worldwide shortage of application developers, every developer hour must be turned to productive work. The .NET Framework development team focused on eliminating as much programming “plumbing” as possible, leaving the developer free to focus on writing business logic. So, for example, the .NET Framework includes time-saving features such as easy-to-use automatic transactions, automatic memory management, and a rich set of controls that encapsulate many common development tasks.

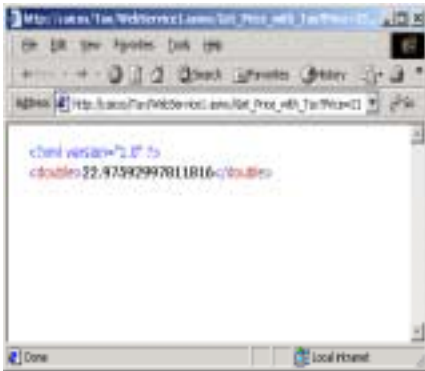
Protecting Investments through Advanced Security

One of the largest concerns with the Internet today is security, and it goes without saying that an Internet development solution needs to have security implemented as an integral component, not layered on top. The security architecture of the .NET Framework is designed from the ground up to ensure that data and applications are protected through a fine-grained, evidence-based security model.

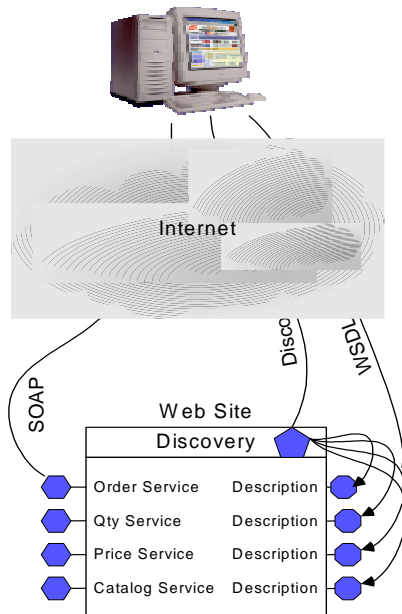
Utilizing Operating System Services

Windows provides the richest services available with any platform; such as comprehensive data access, integrated security, interactive user interfaces, mature component object model, transaction processing monitors and world-class message queuing. The .NET Framework needed to take advantage of this richness and expose it in an easy to use way.

THE XML WEB SERVICES PROGRAMMING MODEL



Example of XML being returned from an XML Web Service



View of accessing an XML Web Service

Core XML Web Services Technologies SOAP Family

At the core of solving the challenges of application integration and delivering software as a service are XML Web services. XML Web services provide a simple, flexible, standards-based model for binding applications together over existing Internet infrastructure. Web applications can be easily assembled with new or existing services, irrespective of the platform, development language, or object model used to implement any of the constituent services or applications.

The key to making XML Web services work across the Web and its heterogeneous infrastructure is to have applications agree on a simple data description format, based on XML. These formats, along with what they do, are described below:

Extensible Markup Language (XML)

XML is the major building block for XML Web services. XML's largest strength is its flexibility to carry any type of data in a platform-independent way. Sample XML being returned from an XML Web service can be seen on the left.

Simple Object Access Protocol (SOAP)

At the lowest level, systems need to speak the same language. In particular, communicating applications need to have a set of rules for how they are going to represent different data types (e.g., integers and arrays) and how they are going to represent commands (i.e., what should be done with the data). Also, the applications need a way to extend this language if they have to. The Simple Object Access Protocol (SOAP), an XML grammar, represents one common set of rules about how data and commands will be represented and extended. SOAP has been accepted by the W3C for standardization.

Web Service Description Language (WSDL)

Once applications have general rules for how they will represent data types and commands, they need a way to describe the specific data and commands they accept. It is not enough for an application to say that it accepts integers; somehow, there must be a way to deterministically say that, if you give it two integers, it will multiply them. The Web Services Description Language (WSDL) is an XML grammar that developers and development tools can use to represent the capabilities of an XML Web Service.

SOAP Discovery ("Disco")

Beyond WSDL, you need a set of rules for how to locate an XML Web service's description – where does a tool look to discover an XML Web service's capabilities? The SOAP Discovery specification provides a set of rules for automatically discovering the WSDL description files on a particular Web site.

Universal Description, Discovery and Integration (UDDI)

UDDI is a directory, like a library card catalog, providing a way to locate all kinds of Web Services. There are three parts to the UDDI specification: White pages, which provide business contact information

Yellow pages, which organize Web Services into categories (e.g. "Credit card authorization services")

Green pages, which provide detailed technical information about individual services

The UDDI Business Registry is an implementation of the UDDI specification and is itself a Web Service that uses SOAP over HTTP as its messaging protocol. For more information, see <http://www.uddi.org/>

The .NET Framework implements all of these Web Services standards.

For more information on XML and SOAP see <http://www.w3.org/XML/Activity.html>

.NET FRAMEWORK: OVERVIEW



View of an Automatically Generated Web Service Client

The .NET Framework: The Microsoft XML Web Services Engine

Clearly, there is a considerable amount of infrastructure required to make building XML Web services transparent to developers and users. The Microsoft .NET Framework provides that infrastructure. The .NET Framework supplies an application model and key enabling technologies to simplify the creation, deployment, and ongoing evolution of secure, reliable, scalable, highly available XML Web services while building on existing developer skills.

To the .NET Framework, all components can be XML Web services, and XML Web services are just another kind of component. In effect, the .NET Framework takes the best aspects of COM (the Microsoft Component Object Model) and combines them with the best aspects of loosely coupled XML Web services-style computing. The result is a powerful, productive Web component system that simplifies programmer plumbing, deeply integrates security, introduces an Internet-scale deployment system, and greatly improves application reliability and scalability.

THE .NET FRAMEWORK: Three Parts

The goal of the Microsoft .NET Framework is to make it easy to build XML Web services and applications, but it also has a dramatic effect on every kind of application, from simple client applications to many other kinds of distributed applications.

The .NET Framework consists of three main parts: the common language runtime, a hierarchical set of unified class libraries, and a componentized version of Microsoft Active Server Pages called Microsoft® ASP.NET.

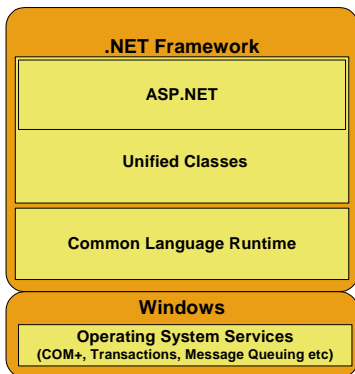
The common language runtime is built on top of operating system services. It is responsible for actually executing the application – ensuring that all application dependencies are met, managing memory, handling security, language integration, and so on. The runtime supplies many services that help simplify code development and application deployment while also improving application reliability.

The developer doesn't actually interact with the runtime, however. Developers use a unified set of classes built on top of the runtime. These classes can be used from any programming language.

As part of these class libraries, the .NET Framework includes a Web application-programming model called ASP.NET that provides higher-level components and services targeted specifically at developing XML Web services and applications.

The Common Language Runtime

The common language runtime is a high performance execution engine. Code



The .NET Framework from the application view



Common Language Runtime

that targets the runtime and whose execution is managed by the runtime is referred to as *managed* code. Responsibility for tasks such as creating objects, making method calls, and so on is delegated to the common language runtime, which enables the runtime to provide additional services to the executing code.

Despite its name, the common language runtime actually has a role in both a component's development time and execution time experiences.

While the component is running, the runtime provides services such as memory management (including garbage collection), process management, thread management, security enforcement, as well as satisfying any dependencies that the component may have on other components.

At development time, the runtime's role changes slightly. Because it automates so much (e.g., memory management), the runtime makes the developer's experience very simple. In particular, features such as lifetime management, strong type-naming, cross-language exception handling, delegate-based event management, dynamic binding, and reflection dramatically reduce the amount of code a developer must write in order to turn business logic into reusable components.

Runtimes are nothing new for languages: virtually every programming language has a runtime. Visual Basic has the most well known runtime (the aptly-named VBRUN), but Visual C++[®] has one (MSVCRT), as do Visual FoxPro[®], JScript[®], SmallTalk, Perl, Python, Haskell, and Java. The critical role of the common language runtime, and what really sets it apart, is that it provides a *unified* runtime environment across all programming languages.

The key features of the runtime include a common type system (enabling cross-language integration), self-describing components, simplified deployment and versioning, and integrated security services.

Common Type System and Multi-Language Integration. The runtime makes use of a new common type system capable of expressing the semantics of modern programming languages. The common type system defines a standard set of data types and rules for creating new types. The runtime understands how to create and execute these types. Compilers for the .NET Framework use runtime services to define data types, manage objects, and make method calls instead of using tool- or language-specific methods.

The result of having the common type system is deep multi-language integration. Code written in one language can inherit implementation from classes written in another language; exceptions can be thrown from code written in one language and caught in code written in another; and operations such as debugging and profiling work seamlessly regardless of the languages used to write the code. This means that developers no longer need to create different versions of their reusable libraries for each programming language or compiler, and developers using class libraries are no longer limited to libraries developed for the programming language they are using.

Metadata and Self-Describing Components. The .NET Framework enables the creation of self-describing components, which simplifies development and deployment and improves system reliability. Self-description is accomplished through metadata – information contained in the binary that supplements the executable code, providing details about dependencies, versions, and so on. The metadata is packaged together with the component it describes resulting in self-describing components.

A key advantage of self-describing components is that you do not need any other files in order to use a component. This is a contrast with typical application development today, which requires separate header files for class definitions, separate interface description language (IDL) files, separate type libraries, and separate proxies and stubs. Since the metadata is generated from the source code during the compilation process, and stored with the executable code, it is never out of sync with the executable.

Because each application contains a full description of itself, the runtime can dynamically assemble a cache of information about the components installed on a system. If that cache becomes damaged somehow, for example, the runtime can rebuild it without the user even knowing. In addition to solving development challenges, self-description eliminates the dependency on the Windows Registry for locating components. A benefit of not relying on the Windows registry is the ability to do No-touch deployment.

“No-touch” Deployment. This is the ability to simply copy a file to a target machine and have it run...No registration needed. In addition to elimination of registry dependencies the .NET Framework includes other deployment advances that virtually eliminate “DLL Hell” – the phenomenon in which a shared library is becomes out of sync from the applications that are trying to access it. “Side by side” deployment now allows multiple versions of the same named libraries to co-exist without conflict. The .NET Framework includes a very strong internal naming system that makes it much harder for two libraries that have the same file name to be mistaken for each other. If a new application overwrites a shared library, an existing application that cannot use the new shared library can actually repair itself. The next time the existing application starts up, it will check its shared files. If it finds that one has changed and that the changes are incompatible, it can ask the runtime to fetch a version it knows it can work with. Because of the security system, the runtime can do so safely, and the application can repair itself.

Evidence-based Security. The .NET Framework takes a major step forward in security by introducing a fine-grained evidence-based security system. This security system now gives the developer and administrator a wide range of privileges that they can grant (not just “on” or “off”). In addition, it enables those privileges to be applied based on key aspects of the code itself, including the origin of the code and digital signatures.

Beyond that, since the common language runtime is used to load code, create objects, and make method calls, the runtime can actually perform low-level

security checks and enforce security policy as code is loaded and executed.

This design ensures that unauthorized users cannot access resources and code cannot perform unauthorized actions, which improves overall system safety and reliability.

What the Programmer Sees: The Unified Classes

The .NET Framework's classes provide a unified, object-oriented, hierarchical, and extensible set of class libraries ("APIs") that developers can use from the languages they are already familiar with.

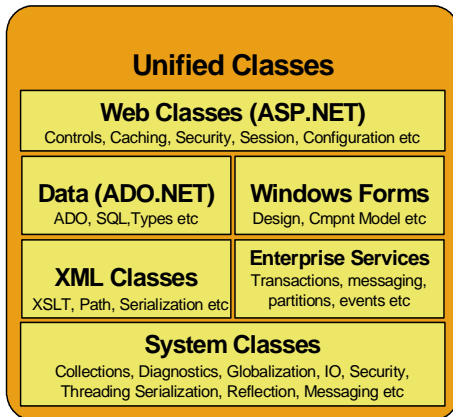
Today, Visual C++ developers use the Microsoft Foundation Classes, Visual J++ developers use the Windows Foundation Classes, and developers using Visual Basic use the Visual Basic framework. Simply put, the .NET Framework's classes unify these different classes, creating a superset of their features. The result is that developers no longer have to learn multiple object models or class libraries. By creating a common set of APIs across all programming languages, the .NET Framework enables cross-language inheritance, error handling, and debugging. In effect, all programming languages, from JScript to C++, become equal, and developers are free to choose the right language for the job.

The .NET Framework provides classes that can be called from any programming language. These classes comply with a set of naming and design guidelines to further reduce the learning curve for developers. Some of the key class libraries are shown in the figure.

The .NET Framework includes a base set of class libraries that developers would expect in any standard library, such as collections, input/output, data type, and numerical classes. In addition, there are classes that provide access to all of the operating system services such as graphics, networking, threading, globalization, cryptography, data access, classes that development tools can use, such as debugging. As well as a set of classes that supply the services necessary for building enterprise scale applications such as transactions, events, partitions and messaging.

Although this Reviewers Guide has focused on the Microsoft .NET Framework as a way to develop XML Web services and Web applications, the unified programming classes also offer full support for developing more traditional Windows-based applications (of course, these applications can use XML Web services, too).

Developers writing client applications for Windows can use the Windows Forms (System.Windows.Forms) classes to take advantage of all the rich user interface features of Windows, including existing ActiveX® controls and new features of Windows 2000, such as transparent, layered, and floating windows. Developers will find the Windows Forms programming model and design-time support very intuitive, given their similarities to existing Windows-based forms packages.



The .NET Framework Unified Classes

Web Applications: ASP.NET

A set of classes within the unified class library, ASP.NET provides a Web application model in the form of a set of controls and infrastructure that make it simple to build Web applications.

ASP.NET comes with a set of server-side controls (sometimes called Web Forms) that mirror the typical HTML user interface widgets (including listboxes, text boxes, and buttons), and an additional set of Web controls that are more complex (such as calendars and ad rotators). These controls actually run on the Web server and project their user interface as HTML to a browser. On the server, the controls expose an object-oriented programming model that brings the richness of object-oriented programming to the Web developer.

One important feature of these controls is that they can be written to adapt to client-side capabilities; the same pages can be used to target a wide range of client platforms and form factors. In other words, Web forms controls can "sniff" the client that is requesting a page and return an appropriate user experience—WML for phones or HTML 3.2 for a down-level browser and Dynamic HTML for Internet Explorer 5.5.

ASP.NET also provides features such as cluster session state management and process recycling, which further reduce the amount of code a developer must write and increase application reliability.

ASP.NET uses these same concepts to enable developers to deliver software as a service. Using ASP.NET XML Web services features, ASP.NET developers can simply write their business logic and the ASP.NET infrastructure will be responsible for delivering that service via SOAP and other public protocols..

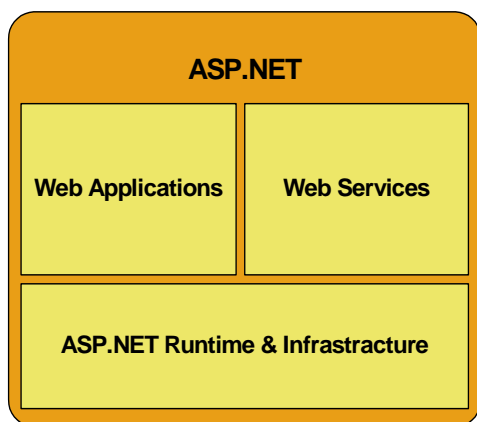
ASP.NET works with all development languages and tools (including Visual Basic, C++, C#, and JScript).

Inside ASP.NET

At the core of ASP.NET is its HTTP runtime (different from the common language runtime), a high-performance execution engine for processing HTTP commands. The HTTP runtime is responsible for processing all incoming HTTP requests, resolving the URL of each request to an application, and then dispatching the request to the application for further processing. The HTTP runtime is multithreaded and processes requests asynchronously, which means it cannot be blocked by bad application code from processing new requests. Furthermore, the HTTP runtime has a resilient design so it is engineered to automatically recover from access violations, memory leaks, deadlocks, and so on.

Updating Applications. ASP.NET uses the Microsoft .NET Framework deployment technologies, thus gaining all its benefits such as XCOPY deployment and side-by-side deployment of applications.

Another major benefit of ASP.NET is support for live updating of applications.



A view of ASP.NET

An administrator does not need to shut down the Web server or even the application to update application files: application files are never locked, so they can be overwritten even when the application is running. When files are updated, the system gracefully switches over to the new version.

Extensible. Within an ASP.NET application, HTTP requests are routed through a pipeline of HTTP modules, ultimately to a request handler. HTTP modules and request handlers are simply managed .NET classes that implement specific interfaces defined by ASP.NET. This modular architecture makes it very easy to add services to applications: just supply an HTTP module. For example, security, state management, and tracing are implemented as HTTP modules by ASP.NET. Higher-level programming models, such as XML Web services and Web Forms, are also implemented as request handlers. An application can be associated with multiple request handlers—one per URL—but all HTTP requests in a given application are routed through the same HTTP modules.

State Management. The Web is a fundamentally stateless model with no correlation between HTTP requests. This can make writing Web applications difficult, since applications usually need to maintain state across multiple requests. ASP.NET enhances the state management services introduced by ASP to provide three types of state to Web applications: application, session, and user. ASP.NET session state is stored in a separate process and can even be configured to be stored on a separate machine or persisted to a SQL Server Database. This makes session state scalable even when an application is deployed across the largest Web farms..

User state resembles session state, but generally does not time out and is persisted. Thus user state is useful for storing user preferences and other personalization information. All the state management services are implemented as HTTP modules, so they can be added, extended, or even removed from an application's pipeline easily. If additional state management services are required beyond those supplied by ASP.NET, they can be provided by a third-party module.

Caching. The ASP.NET programming model provides a cache API that enables programmers to activate caching services (on enterprise software) to improve performance. An output cache saves completely rendered pages, and a fragment cache stores partial pages. Classes are provided so applications, HTTP modules, and request handlers can store arbitrary objects in the cache as needed.

Aggressive caching capabilities will be provided as part of the ASP.NET. ASP.NET is designed to provide a robust Web application environment capable of running mission-critical projects for long periods of time.

Even though it's not yet released, ASP.NET is already delivering some significant improvements in Web applications – performance improvements of up to 3X over existing ASP-based applications, and even more dramatic productivity improvements.

Benefit Summary

- **Use any programming language**
- **Take advantage of industry-leading tools**
- **Developers write less code due to greater componentization and reduction of “plumbing”**
- **Employ Windows application servicesMulti-language integration enables developers to take existing skills to new challenges**
- **Integrated XML Web services technologies make it simple to build next generation applications.**
- **Use COM+ Services for easy access to advanced scalability and reliability features**

Use any programming language

The .NET Framework enables developers to use any programming language, and for applications written in any programming languages to integrate deeply with each other, enabling current development skills to go forward without retraining.

Objects written on top of the .NET Framework can integrate with each other regardless of the language in which they were created because they are built on top of a common type system. The .NET Framework’s multi-language integration includes:

- Calling methods on other objects
- Inheriting implementations from other objects
- Passing instances of a class to other objects
- Using a single debugger across multiple objects
- Trapping errors from other objects

There are several scenarios where cross-language integration is useful. For example, development projects gain access to a larger skills base – they can choose from developers skilled in any programming language and join them together on the same team. Alternately, if you are writing components for a distributed Web application, it would be helpful to know that no matter what language you choose to write your components in, they can interact closely with each other and with components supplied by other developers.

Take advantage of industry-leading tools

The .NET Framework is designed for development tools and it includes many features specifically targeted as improving the quality of the tools that use it such as integrated debug and profiling. Microsoft Visual Studio .NET, the industry-leading development environment, is built on top of and designed to take advantage of the .NET Framework.

Developers write less code

The .NET Framework uses a highly componentized, plumbing-free design that enables developers to focus on writing business logic. Developers don’t need to write IDL or Registry code, and ASP.NET, for example, includes dozens of controls that encapsulate common programmer tasks such as a shopping cart

The .NET Framework decreases the amount of code the developer has to write. This is done through many mechanisms such as self describing metadata, automatic memory management, a rich set of controls, simplified programming classes, and the capability to automatically transform any application method into an XML Web service with the addition of a single keyword.

Decreasing the plumbing. With most development environments today, the developer has to cope with infrastructure overhead – details that are required for building an application that aren’t actually business logic. The .NET

Framework reduces this through many mechanisms such as automatic memory management, metadata, a rich set of controls, and the capability to automatically transform any application method into an XML Web service with the addition of a single keyword.

Automatic Memory Management. Many of the problems associated with modern applications can be attributed to memory leaks; in fact keeping track of the memory being used can be a very daunting task. The .NET Framework solves this problem with a high-performance, generational memory manager that is responsible for freeing unused memory efficiently and without imposing a high overhead on application performance.

Metadata. Metadata enables developers to write less code by eliminating the need for separate interface definition language files (IDL files). This is possible because the objects themselves maintain metadata. This metadata describes information about the methods, fields, properties, and events associated with a class.

Controls. The controls in the .NET Framework offer code savings. These savings occurs in two primary areas: the adaptive user interface features and server-side controls.

Automatic Browser Detection. Automatic browser detection is the capability for the runtime to automatically sense the capabilities of the client and render suitable HTML. So the HTML rendered to achieve a consistent look of a Web page for down-level browser software and up-level browser software such as Internet Explorer 5.5 client may look radically different but the ASP.NET code is exactly the same.

Server-side Controls. Server side controls assist developers by separating the code and the content, enabling developers to concentrate on the business logic they are trying to write.

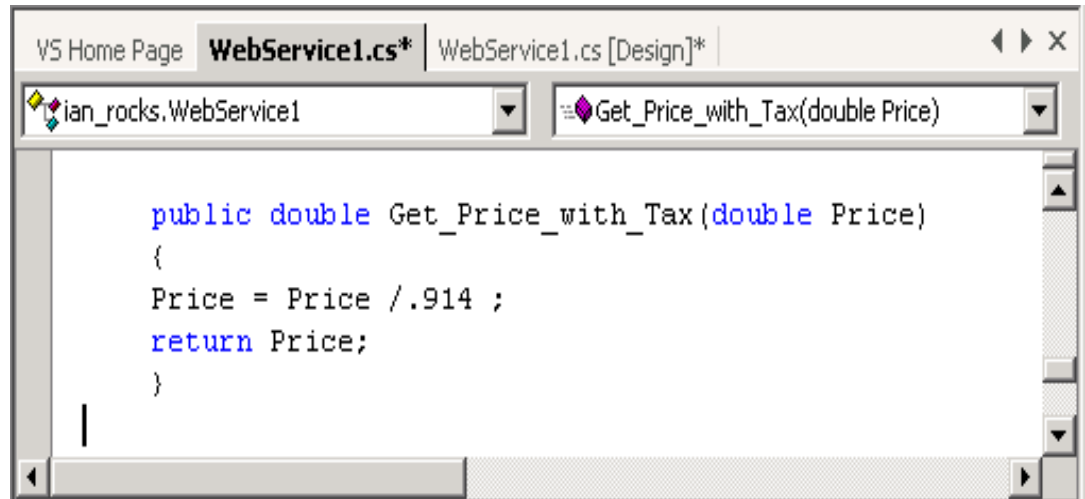
Employ Windows 2000 Applications Services

Windows 2000 has the most advanced application services available: the fastest transaction monitor and message queuing engine, the most advanced data access subsystem, the best Web server. The .NET Framework takes advantage of these. For example, .NET Framework transactions are COM+ transactions, and all the new COM+ capabilities in WindowsXP and Windows .NET Server can be used from the .NET Framework.

Create XML Web services Easily

Taking advantage of the Internet using COM component interfaces was made simple in 1996 with the release Active Server Page (ASP) scripts running under Windows 2000 Server's Internet Information Services (IIS). For example, most developers could easily update a database from a Web page using ADO, Visual Basic Scripting Edition (VBScript) and ASP. As outlined in the ASP.NET section of this guide, Microsoft is making doing this even easier and more reliable. With

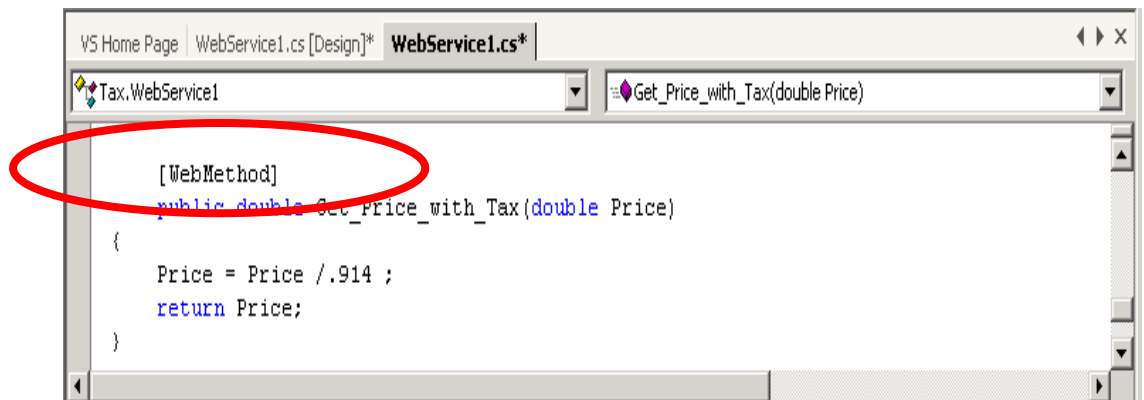
the .NET Framework, all objects can be made Internet-accessible by simply marking them with the keyword: "WebMethod."



The screenshot shows a Visual Studio window with the file 'WebService1.cs' open in Design view. The dropdown menu shows 'ian_rocks.WebService1' and the selected method is 'Get_Price_with_Tax(double Price)'. The code editor contains the following C# code:

```
public double Get_Price_with_Tax(double Price)
{
    Price = Price /.914 ;
    return Price;
}
```

*C# Function Made
Available as an XML
Web service*



The screenshot shows the same Visual Studio window, but now the dropdown menu shows 'Tax.WebService1'. The code editor now includes the '[WebMethod]' attribute above the method signature, which is circled in red:

```
[WebMethod]
public double Get_Price_with_Tax(double Price)
{
    Price = Price /.914 ;
    return Price;
}
```

The important thing to note in the first example is that the function was not available to the Internet until the keyword "[WebMethod]" was added.

But this is only half the solution; the *WebMethod* keyword gives programmers the capability to expose functionality to the Internet, but how does a developer use these functions? The .NET Framework supplies the utility *WebServiceUtil.exe* that takes the path to a WSDL file and a language (C#, Visual Basic or JScript) as input arguments. The utility then generates a single source file as output. This file contains a proxy class, defined using the language specified, that exposes methods (both synchronous and asynchronous) for each of the methods exposed by the XML Web service. Each proxy method contains the appropriate network invocation and marshalling code necessary to invoke and receive a response from the remote service. Visual Studio .NET makes this even easier by doing this through an

explorer interface (please see the Visual Studio .NET chapters for more information on the Visual Studio Server Explorer.)

Use COM+ Services

Introduced in 1992, the Microsoft Component Object Model (COM) is the most popular object model for building reusable components, with more than 10,000 components commercially available. Over time, COM has evolved from a technology for embedding one document inside another (OLE), to a technology for distributing self-contained applications (ActiveX), to a technology for building and running middle-tier business logic (COM+).

With so many years of experience, Microsoft architects undertook the challenge of making COM development easier. Visual Basic began to show how to accomplish this in 1995. The .NET Framework takes that simplification to its next step, automating features of COM such as reference counting, interface description, and registration.

The .NET Framework natively supports COM. In fact, a COM developer using the Microsoft Visual Studio 6.0 development system could call a .NET Framework component and, to the developer, it would look like a COM component, complete with IUnknown. Conversely, a .NET Framework developer using Visual Studio .NET would see a COM component as a .NET Framework component.

COM+ is COM combined with MTS, DCOM, and other component services such as Partitions. COM+ provides a set of middle-tier-oriented services. In particular, COM+ provides process management and database and object connection pooling. In future versions, it will also provide stronger process isolation designed for application service providers – a feature called partitioning.

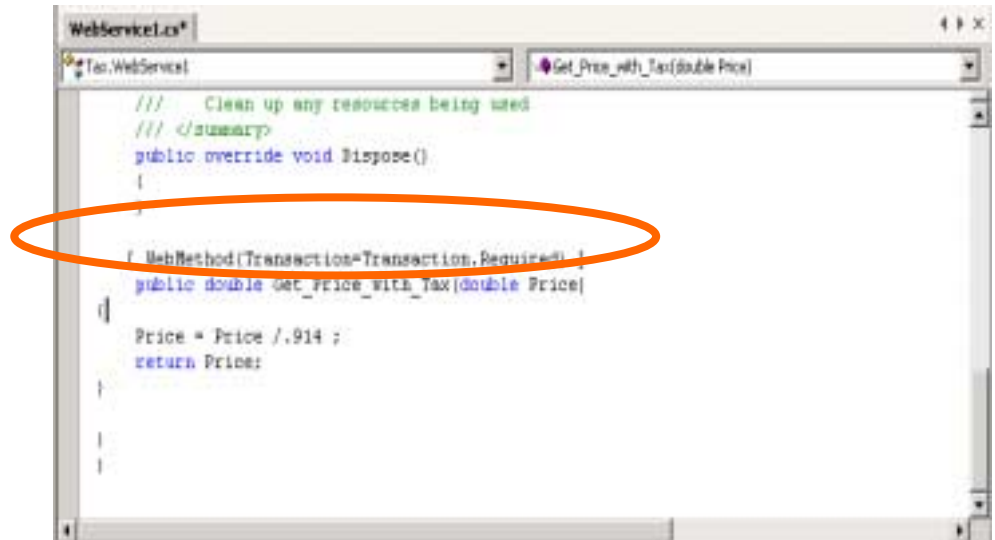
The COM+ services are primarily oriented toward middle-tier application development and focus on providing reliability and scalability for large-scale, distributed applications. These services are complementary to the programming services provided by the .NET Framework. The .NET Framework provides direct access to these component services.

You can think of COM+ providing component services for building scalable, reliable distributed middle tier applications. The .NET Framework provides language services that simplify and speed up application development and deployment.

Automatic Transaction Support. One of the most critical features of COM+ that developers will want to use from the .NET Framework is COM+ Transactions. It is easy to build applications that need transaction support in the .NET Framework through simple keywords and settings. In fact, the .NET Framework takes advantage of the COM+ transaction infrastructure to provide both manual and automatic distributed transactions.

The beauty of the integration between the .NET Framework and COM+ transactions is that .NET Framework developers can use the high-performance COM+ transaction environment without needing to learn COM+. For example, you can simply copy your .NET Framework components that need COM+ Services such as transactions into a directory, and the .NET Framework will dynamically add it to the COM+ transaction monitor.

*C# Function Using
Transactions through a
Keyword*



```
WebService1.cs*
*Tax.WebService*
Get_Price_with_Tax(double Price)

/// Clean up any resources being used
/// </summary>
public override void Dispose()
{
}

[ WebMethod(Transaction=Transaction.Required) ]
public double Get_Price_with_Tax(double Price)
{
    Price = Price /.914 ;
    return Price;
}

}
```

AGILITY TO SOLVE TODAY'S BUSINESS PROBLEMS

Benefit Summary

- **Deliver on Software as a Service**
- **Supporting ECMA standards for C# and CLI**
- **Extend all existing software transparently**
- **Access databases easily with ADO.NET**
- **Reduce total cost of ownership around application deployment**
- **Protect investments with advanced security (Moved this to Improved Operations underneath Evidence-based security)**
- **Next-generation data access designed for Web-scale applications**

Deliver on Software as a Service

The .NET Framework was built for delivering software as a service, so it is built on XML and the SOAP family of integration standards. Simply annotate method calls and the .NET Framework turns them into full XML Web services. SOAP is being standardized by the W3C.

Supporting ECMA standards for C# and CLI

Standards are core to delivering software as a service. Consequently, Microsoft has submitted the specifications for the C# programming language and a subset of the .NET Framework called the Common Language Infrastructure (CLI) to ECMA and ECMA is standardizing them. These specifications are a collaboration of six other ECMA partners including Hewlett-Packard Co and Intel. The standardization process is currently about half way completed

Device support for broad reach

The Microsoft .NET Compact Framework is a version of the .NET Framework for rapidly building, and securely deploying and running distributed XML Web services and applications on smart devices such as cellular telephones, enhanced televisions, and PDAs. It provides a highly productive, standards-based, multi-language environment for integrating existing investments with next-generation applications and services as well as the agility to solve the challenges of deployment and operation of Internet-scale applications. The .NET Compact Framework consists of three main parts: the common language runtime, a hierarchical set of unified class libraries, and a set of profiles for individual categories of smart device.

Extend all existing software transparently

The .NET Framework is designed to integrate with your existing software, enabling you to take advantage of all your existing development investments without replacing them. For example, all your existing COM components are .NET Framework components automatically, and any .NET Framework component you create is also a COM component.

Access databases easily with Microsoft ADO.NET

Microsoft® ADO.NET provides a productive interface to any database and is designed for today's loosely coupled style of data access that Web applications in particular use. For easy interoperability, ADO.NET uses XML as its native data format. It builds on the huge library of ODBC drivers already available.

Web Data Access

Nearly all applications need to query or update persisted data, whether in simple files, relational databases, or any other type of store. To fulfill this need, the .NET Framework includes ADO.NET, a data access subsystem optimized

for N-tier environments and is interoperable with XML and XML documents. ADO.NET is designed for loosely-coupled environments. As the name implies, ADO.NET evolved from ADO, the ActiveX® Data Objects.

ADO.NET is designed to provide data access services for scalable Web-based applications and services. ADO.NET provides high-performance stream APIs for connected, as well as a disconnected data model more suitable for returning data to Web applications.

As you develop applications, you will have different requirements for working with data. In some cases, you might simply want to display data on a form. In other cases, you might need to devise a way to share information with another company.

No matter what you do with data, there are certain fundamental concepts that you should understand about the data approach in the .NET Framework. You might never need to know some of the details of data handling — for example, you might never need to directly edit an XML file containing data — but it is very useful to understand the data architecture, what the major data components are, and how the pieces fit together.

Disconnected Data Architecture

In traditional two-tier applications, components establish a connection to a database and keep it open while the application is running. For a variety of reasons, this approach is impractical in many applications:

- **Open database connections take up valuable system resources.** The overhead of maintaining these connections impacts overall application performance.
- **Applications that require an open database connection are extremely difficult to scale.** An application that might perform acceptably with four users will likely not do so with a hundred. Web applications in particular need to be easily scalable, since traffic to a Web site can go up by orders of magnitude in a very short period.
- **In Web applications, the components are inherently disconnected from each other.** The browser requests a page from the server; when the server has finished processing and sending the page, it has no further connection with the browser until the next request. Under these circumstances, maintaining an open connection to a database is not viable, since there is no way to know whether the data consumer (the client) requires further data access.
- **A model based on connected data can make it difficult to share data between components, especially components in different applications.** If two components need to share the same data, both have to be connected, or a way must be devised for the components to pass data back and forth.

For all these reasons, data access in ADO.NET is designed around a disconnected architecture. Applications are connected to the database only

long enough to fetch or update the data. Because the database is not hanging on to connections that are largely idle, it can service many, many more users.

Data is cached in datasets

Far and away the most common data task is to retrieve data from the database and do something with it: display it, process it, or send it to another component. Very frequently, the application needs to process not just one record, but a set of them: a list of customers or today's orders, for example. Often the set of records that the application requires comes from more than one table: my customers and all their orders; all authors named "Smith" and the books they've written; and other, similar, sets of related records.

Once these records are fetched, the application typically works with them as a group. For example, the application might allow the user to browse through all the authors named "Smith" and examine the books for one Smith, then move to the next Smith, and so on.

In a disconnected data model, it's impractical to go back to the database each time the application needs to process the next record. (Doing so would undo much of the advantage of working with disconnected data in the first place.) The solution, therefore, is to temporarily store the records retrieved from the database and work with this temporary set.

This is what a dataset is. A dataset is a cache of records retrieved from the database. It works like a virtual data store – it includes one or more tables based on the tables in the actual database (or databases), and it can include information about the relationships between those tables and constraints on what data the tables can contain.

The data in the dataset is usually a much-reduced version of what's in the database. However, you can work with it in much the same way you do the real data. While you are doing so, you remain disconnected from the database, which frees it to perform other tasks.

Of course, you often need to update data in the database (although not nearly as often as you retrieve data from it). You can perform update operations on the dataset, and these are written through to the underlying database.

An important point is that the dataset is a passive container for data. To actually fetch data from a database and (optionally) write it back, you use a data adapter. A data adapter contains the instructions for how to populate a single table in the dataset and how to update the corresponding table in the database. The instructions are methods that encapsulate SQL statements, such as a reference to a stored procedure. Thus, the **Fill** method may invoke a SQL statement such as `SELECT au_id, au_lname, au_fname FROM authors` that runs whenever the method is called.

Data is persisted as XML

Data needs to be moved from the data store to the dataset, and from there, to various components. In ADO.NET, the format for remoting data is XML.

When data needs to be persisted outside of the database (for example, into a file), it is stored as XML. If you have an XML file available, you can use it like any data source and create a dataset out of it.

In fact, in ADO.NET, XML is the fundamental format for sharing data. When you share data, the ADO.NET APIs automatically create XML files or streams out of information in the dataset and send them to another component. The second component can invoke similar APIs to read the XML back into a dataset.

Why XML? There are several reasons:

- **XML is an industry-standard format.** This means that your application data components can exchange data with any other component in any other application, as long as that component understands XML. Many applications are being written to understand XML, which provides an unprecedented level of exchange between disparate applications.
- **XML is text-based.** The XML representation of data uses no binary information, which enables XML to be sent via any protocol, such as HTTP. Most firewalls block binary information, but by formatting information in XML, components can still easily exchange the information.
- **Interoperability.** ADO.NET enables easy creation of custom XML documents through the use of XSD schemas. The resulting XSD schemas format the XML specific for your use.

Do you need to know XML in order to share data in ADO.NET? No. ADO.NET automatically converts data into and out of XML as needed; you interact with the data using ordinary programming methods.

Benefits of ADO.NET

ADO.NET offers several advantages over other data access solutions. These benefits fall into the following categories:

Leverage data through interoperability

ADO.NET applications can take advantage of the flexibility and broad acceptance of XML. Because XML is the format for transmitting datasets among components and across tiers, any component that can read the XML format can process an ADO.NET dataset. As an industry standard, XML was designed with exactly this kind of interoperability in mind.

Simplified code through typed programming

Programmers can manipulate objects of the ADO.NET model through typed programming. Typed programming is programming in which the types of things that are important to users are recognized by the programming environment or programming language itself.

For example, consider the following line of code, using generic (non-typed) programming:

```
If TotalCost >  
Table("Customer").Column("AvailableCredit")
```

The code contains words such as "Customer" and "Available Credit" that are interesting to the end user. The code also contains the words "Table" and "Column".

In a nutshell, typed programming is a style of programming in which the end-user words figure prominently. For example, consider the following line of code using typed programming in ADO.NET:

```
If TotalCost >  
DataSet1.Customer("Jones").AvailableCredit
```

This line of code is equivalent to the earlier line using non-typed programming. In the typed code, the code is easier to read: A business analyst with little or no programming experience can grasp the meaning of the condition being tested without having to filter out the programmer vocabulary in the conventional line of code.

The typed code is also easier to write, because statement completion is provided. For example, "AvailableCredit" is among the list of choices for completing the following statement:

```
IF TotalCost > Customer.
```

In addition, the typed code is safer, because it provides for the compile-time checking of types. For example, suppose that `AvailableCredit` is expressed as a currency value. If the programmer erroneously assigns a string value to `AvailableCredit`, the typed environment would report the error to the programmer during compile time. In a weakly typed programming environment, the programmer would not learn of the error until run time.

Scalability

Because the Web can vastly increase the demands on your data, scalability has become critical. Internet applications have a limitless supply of potential users. Although an application might serve a dozen users well, it might not serve hundreds — or hundreds of thousands — equally well. An application that consumes resources such as database locks and database connections will not serve high numbers of users well, because the user demand for those limited resources will eventually exceed their supply.

Because any ADO.NET application employs disconnected access to data, it does not retain database locks or active database connections for long durations and offers performance advantages.

IMPROVED OPERATIONS

Benefit Summary

- Evidence-based security
- Simplify application deployment
- Run more reliable applications
- Improve performance

Evidence-based security

The .NET Framework security system provides fine-grained, method-level control over what applications can and can't do based on who wrote the code, what it's trying to do, where it was installed from, and who is trying to run it.

Internet Capable Security

The .NET Framework security infrastructure encompasses features to handle both authentication of users and code access security that enforces security permissions on code based on trust policy. An extensible library of cryptographic functions provides easy access to hashing and encryption from managed code, including digital signatures for XML. All existing security mechanisms in the platform exist with and provide the foundation for .NET Framework security to build upon.

Code Access Security and Role-Based Security. The .NET Framework offers security through two high-level categories: *code access security* and *role based security*. Code access security is the mechanism by which developers can specify the level of access their code should have to resources and operations. Role based security is used for controlling permissions based on the user identity.

Code access security is deeply integrated into the .NET Framework, and provides security enforcement of different levels of trust on different code – even different pieces of code within the same running application. For example, code from the Internet should be trusted less than code from a reliable vendor. Many security problems result from less trusted code getting outside the confines meant to secure it. Since the runtime has control of execution of all managed code, it continues to enforce restrictions for anything the less trusted code might do.

Code access security is built on an evidence-based security policy system that grants to code the appropriate permissions. A permission object represents the right to do some operation, for example, to write a file, to display on the screen, and so forth. The *evidence* for any code – its location, digital signature, and so forth – is presented to the security policy system for it to make a trust decision based on policy established by the administrator. If an application attempts to perform a protected operation, a security check will demand that the code and all its callers have the necessary permission in order to allow it. The check of all callers is important to prevent less trusted code from somehow tricking more trusted code into doing a privileged operation. The .NET Framework can force managed code to pass a process called verification to enforce security on it. Verification ensures that the code uses only well-defined interfaces in interacting with other objects, preventing access to unauthorized memory.

Role-based security is based on two fundamental concepts: authentication and

authorization. Authentication is the process of validating a set of user credentials against an authority. If the credentials are valid, that user is said to have an identity. Authorization is the process of using that identity to grant access to or protect a resource.

Applications can authenticate users with any authentication protocol, including Basic, Digest, Microsoft Passport, Integrated Windows authentication (formerly known as Windows NT[®] LAN Manager or NTLM), or form-based authentication. The application program is written the same way no matter what type of authentication is used. Moreover, additional user-defined authentication providers can be plugged into this uniform architecture if needed, so for example a customer database with user names and passwords can easily be used as a new authentication provider.

Cryptography APIs. A cryptographic library provides easy-to-use hash, encryption, decryption, and random number generation. Encryption support for several symmetric and asymmetric algorithms is included. In addition, Microsoft is tracking the emerging XML digital signature standards work and will have methods to sign and verify signed XML.

Simplify application deployment

With the .NET Framework metadata technology, installing applications is as easy as copying them into a directory and eliminate “DLL Hell.” The .NET Framework is even capable of self-healing when applications are damaged.

Application deployment is greatly simplified with the .NET Framework. One of the largest challenges faced by developers and administrators (and ultimately users) is versioning. If your system works fine today and you install a new application and suddenly nothing works any more, it’s very often because the new application overwrote some shared library and (more often than not) fixed some bug that existing applications were relying on. This is so frequent, there’s even a name for it: DLL Hell.

The .NET Framework includes a couple of advances that virtually eliminate DLL Hell. First, it includes a very strong internal naming system that makes it much harder for two libraries that may even have the same file name to be mistaken for each other. Beyond that, there’s a new feature called “side by side” deployment. If that new application really does overwrite a shared library, the existing application can actually repair itself. The next time the existing application starts up, it’ll check all its shared files. If it finds that one has changed and that the changes are incompatible, it can ask the runtime to fetch a version it knows it can work with.

Side-by-side deployment is enabled by *assemblies* and *manifests*.

What is an assembly?

An assembly is the primary building block of a .NET application. It is a collection of functionality built, versioned, and deployed as a single implementation unit

(one or multiple files). All managed types and resources are marked either as assessable only within their implementation unit or as exported for use by code outside that unit.

Assemblies are self-describing via their manifest, an integral part of every assembly.

What does the manifest do?

- Establishes the assembly identity, in the form of a text name, version, culture, and if the assembly is to be shared across applications, a digital signature.
- Defines what files (by name and file hash) make up the assembly implementation.
- Itemizes the compile-time dependencies on other assemblies.
- Specifies the types and resources that make up the assembly, including which ones are exported from the assembly.
- Specifies the set of permissions required for this assembly to run properly.

This information is used at runtime to resolve references, enforce version binding policy, and validate the integrity of loaded assemblies. The runtime can determine and locate the assembly for any running object, since every type is loaded in the context of an assembly. Assemblies are also the unit at which code access security permissions are applied. The identity evidence for each assembly is considered separately when determining what permissions to grant the code it contains.

Put simply, assemblies contain information about what the things inside depend upon. The runtime is responsible for making sure these dependencies are fulfilled, including fetching necessary components from approved sites (a technique that uses the .NET Framework security system). Because the runtime is responsible for actually managing applications as they run, the .NET Framework is even able to run two versions of the same component side by side. The self-describing nature of assemblies also helps makes zero-impact install and No-touch deployment feasible.

It has always been possible for multiple copies of a software component to reside on the same system. In the past, however, only one of these copies can be registered with the operating system or loaded for execution - the policy for locating and loading components is global to the system. Through side-by-side deployment, the .NET Framework adds the infrastructure necessary to support per-application policies that govern the locating and loading of components.

Application configuration information defines where to look for assemblies, thus the runtime can load different versions of the same assembly for two different applications that are running concurrently. This eliminates issues that arise from incompatibilities between component versions, improving overall system stability. If necessary, administrators can add configuration information, such as a different versioning policy, to assemblies at deployment time, but the original

information provided at build time is never lost.

Because assemblies are self-describing, no explicit registration with the operating system is required. Application deployment can be as simple as copying files to a directory tree. Configuration information is stored in XML files that can be edited by any text editor.

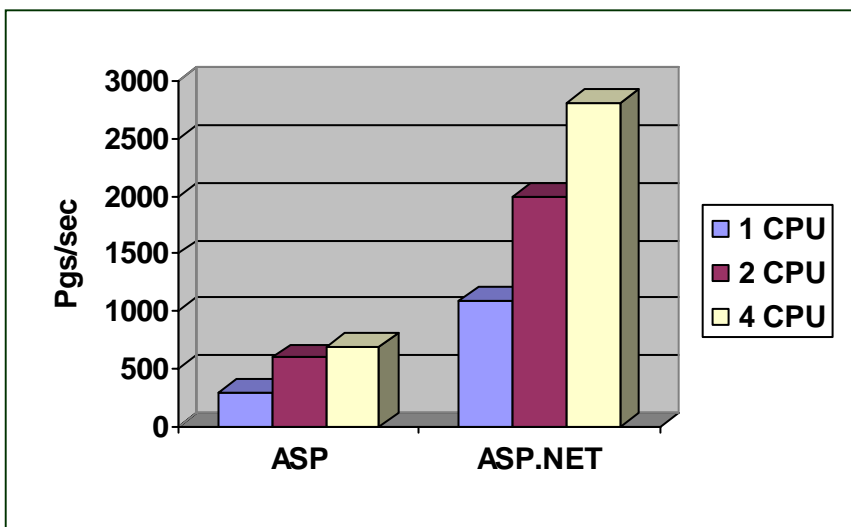
Run more reliable applications

The .NET Framework includes technologies to make applications more reliable. For example, memory, threads, and processes are managed by the .NET Framework to ensure that memory leaks don't occur. And ASP.NET monitors running Web applications, and can automatically restart them at administrator-defined intervals. Many of the reliability features in the .NET Framework come from the application services supplied by the unified classes called Enterprise Services (System.EnterpriseServices). Using the same engine as COM+, these application services provide such features as transactions, messaging, events, partitions, and object request broker

Improve performance

The .NET Framework improves the performance of typical Web applications. ASP.NET includes advanced compilation and caching features that improve performance by a factor of two to three over existing Active Server Pages applications.

Typical web application running ASP versus ASP .NET. Both on the same computer running Windows 2000 and SQL Server 2000



.NET Framework—Features at a Glance	
XML Web Services Standards Support	
Pervasive use of Extensible Markup Language (XML)	XML is a data format for structured document interchange on the Web. The .NET Framework uses XML pervasively from describing objects to security configuration files
Interoperability through SOAP support	The .NET Framework enables developers to expose and consume XML Web services transparently through Simple Object Access Protocol (SOAP), a standard XML grammar for application interoperability.
Easy Description of XML Web Services with the Web Services Description Language	The .NET Framework transparently generates Web Services Description Language (WSDL) descriptions of XML Web services.
Expose Web Services with SOAP Discovery (Disco)	SOAP Discovery provides the mechanism by which XML Web services can be found on a Web server; the .NET Framework provides a transparent way of publishing XML Web services through SOAP Discovery.
Support for UDDI	The .NET Framework uses Universal Description, Discovery and Integration (UDDI) to enable XML Web services which provides the mechanism by which XML Web services can be found on the Internet(In addition to other resources)
Developer Productivity	
Multiple Programming Languages Integration	The .NET Framework provides deep, cross-programming language integration, boosting productivity by enabling developers to extend one programming language's components within another language by way of cross-language inheritance, debugging, and error-handling.
Automatic Versioning	Part of the self-describing nature of every .NET Framework component and application is that each has a unique name that it holds internally. Because applications bind to components' unique names rather than filenames, the .NET Framework runtime is capable of handling the versioning of shared components intelligently, doing away with so-called "DLL Hell."
"No-touch" Deployment	The .NET Framework includes advanced features for deploying applications. By eliminating the need for the Windows Registry, installing an application onto a system is as simple as copying it into a target directory and running it.
Automatic Memory Management	The .NET Framework is a garbage-collected environment. Garbage collection frees applications using .NET Framework objects from the need to explicitly destroy those objects, reducing common programming errors dramatically.
Self-Describing Components	The metadata that every .NET Framework object contains enables the runtime to interrogate objects for data types, functionality, etc., and to verify that the objects are being called correctly, rather than allowing the call to happen and fail after the fact. This feature is called <i>reflection</i> .
Advanced ASP.NET user interface control model and controls	Increases productivity by encapsulating complex interactions in server-side components.

Separation of code and content	Enables developers and content creators to work in parallel by keeping the content in a file separate from the application code.
Deep Platform Integration	The .NET Framework enables developers to use all existing Windows applications and services. This enables developers to use their existing code while taking advantage of the advanced capabilities in the .NET Framework.
Write Less Code	Because the .NET Framework uses highly componentized, plumbing-free design, developers can concentrate on writing business logic rather than plumbing for memory and state management or determine the capability of a client's browser. .
Agility to solve today's business challenges	
Support Public Internet Standards	Standards are core to deliver software as a service. Microsoft submitted the specifications for the C# programming language and the Common Language Infrastructure (CLI) to ECMA for standardization.
Asynchronous Support	The .NET Framework deeply integrates two asynchronous communication technologies for scalability and reliability: SOAP and Microsoft Message Queuing Services (MSMQ).
Transaction Support	Application developers can include the .NET Framework operations within transactions that contain other activities, such as database updates. The .NET Framework supports transactions through Microsoft Transaction Services (MTS) and COM+, and provides a standards-compliant XA interface.
ASP.NET Compilation	Compilation increases performance by compiling pages instead of interpreting them. Supports both pre-compiled applications and on-the-fly-compiled applications.
Process-independent, Web-farmable session state	Increases reliability and scalability by storing session state in a process external to the ASP.NET application so the state can survive application crashes and be referenced from other machines in a web farm.
Automatic ASP.NET session monitoring and restart	Increases reliability by monitoring running ASP.NET applications and even stopping and starting them when necessary.
Universal Data Access with ADO.NET	The .NET Framework includes ADO.NET, a productive interface to any database designed specifically for today's loosely coupled style of data access. ADO.NET provides data access services for scalable Web-based applications and XML Web services, including support for connected, as well as disconnected data models.
Improved Operations	
Evidence-based Security	The .NET Framework's code access security system enables developers to specify the required permissions that their code needs to accomplish work. The common language runtime is responsible for ensuring that these permissions are either met or rejected, depending on evidence including the user identity, code identity, what the code is actually trying to do, where the code originated from and so on

Integrated Windows Authentication	<p>The .NET Framework also integrates with Windows Authentication. Integrated Windows authentication was formerly known as both NT LAN Manager (or NTLM) and Windows NT® Challenge/Response authentication. In integrated Windows authentication, the browser attempts to use the current user's credentials from a domain logon. If those credentials are rejected, integrated Windows authentication will prompt the user for a user name and password by means of a dialog box. When integrated Windows authentication is used, the user's password is not passed from the client to the server. If a user has logged on as a domain user on a local computer, the user won't have to be authenticated again when accessing a network computer in that domain.</p>
--	---

Internet Authentication	<p>Internet users often need different authentication mechanisms. Applications using the .NET Framework can take advantage of and be configured for authentication using a combination of Web Server and .NET Framework authentication providers.</p> <p>Anonymous:</p> <p style="padding-left: 40px;">The server logs on the user with an anonymous or guest account.</p> <p>Cookie:</p> <p style="padding-left: 40px;">Cookie authentication is generally used to refer to a system whereby unauthenticated requests are redirected to an HTML form (using HTTP client-side redirection). The user provides credentials and submits the forms. If the application authenticates the request, the system issues a cookie that contains the credentials in some form or a key for reacquiring the identity. Subsequent requests are issued with the cookie in the request headers and they are authenticated and authorized by an ASP.NET handler using whatever validation method the application desires.</p> <p>Basic:</p> <p style="padding-left: 40px;">Basic authentication is a security mechanism using a standard HTTP mechanism in which user information is sent and received as clear text. It uses a base64 encoded string that contains the user name and password. Passwords and user names are encoded but not encrypted in this type of authentication.</p> <p>Digest</p> <p style="padding-left: 40px;">More advanced than basic authentication, digest authentication sends a hash value, rather than the password, over the network. This method works across proxy servers and other firewalls.</p> <p style="padding-left: 40px;">A challenge-response scheme that challenges using a nonce (a server-specified data string) value. A valid response contains a checksum of the user name, the password, the given nonce value, the HTTP method, and the requested URI.</p> <p>Passport:</p> <p style="padding-left: 40px;">Passport authentication is a centralized authentication service provided by Microsoft that offers a single sign-in and core profile services for member sites.</p> <p>Custom</p> <p style="padding-left: 40px;">The .NET Framework enables programmers to use their own authentication implementations to supply custom authentication schemes; this enables the flexibility to use custom backends or databases.</p>
Improved Performance	<p>The .NET Framework improves the performance of typical Web applications. ASP.NET using the ability to optimize for certain processors and aggressive caching ASP.NET can improve performance in excess of two to three times over existing Active Server Pages applications.</p>

Rock solid Reliability	Achieve enterprise scale reliability with the .NET Framework, through the automated memory management, guaranteed version compatibility, dynamic application recycling and the enterprise services supplied by the COM+ engine.
-------------------------------	---

SUMMARY

Building Next-Generation Web Applications

The .NET Framework is an integral component of the Microsoft .NET Platform, the end-to-end Internet platform for rapidly building and deploying XML Web services and Web applications that integrate customers, businesses and applications. The .NET Framework enables developers to rapidly create XML Web services and Web Applications through the use of developer productivity features such as multiple language support, adherence to public Internet standards, and thru the use of a loosely coupled scalable architecture. While many vendors sell application servers separately, Microsoft delivers a comprehensive set of Web application services fully integrated into the Windows operating systems and available through .NET Framework.

Loosely Coupled Designs Enable Scalability

The core XML Web services technologies upon which the .NET Framework is built are loosely coupled, which means you can change the implementation at either end of a connection and the application will continue working. Technically, this translates to using message-based, asynchronous technology to achieve scalability and reliability, and using Web protocols such as HTTP, SMTP, and, most importantly, XML to achieve universal reach.

Leverages Operating System Services

Because other solutions try to provide identical functionality across many different platforms, these solutions are limited to a *least-common denominator* approach, which means that services provided by the operating system are abandoned and recreated from scratch. In contrast, the .NET Framework design pattern is designed to exploit the power and flexibility of the operating system.

Multi-Language Support

With the .NET Framework developers can take advantage of the programming language they are most comfortable with or use the language that is best suited to the job at hand. Having objects of one language call into objects of another may not be new but the .NET Framework takes this a step further by enabling developers to do cross language implementation inheritance, cross language debugging and do all of this with a single easy to use unified class library.

Standards Based

The .NET Framework is based on Extensible Markup Language (XML) and the Simple Object Access Protocol (SOAP), which are standards overseen by standards bodies. In addition, key parts of the .NET Framework are being submitted to ECMA for standardization. This is much a safer solution than one wholly owned and controlled by a single company.

APPENDIX A FREQUENTLY ASKED QUESTIONS

What is .NET?

Simply put, Microsoft® .NET is Microsoft's strategy for delivering software as a service. For complete information, go to this link:

<http://www.microsoft.com/net/whatis.asp>.

An excerpt from that paper briefly describes the key points of .NET:

- Microsoft .NET platform
Includes .NET infrastructure and tools to build and operate a new generation of services, .NET user experience to enable rich clients, .NET building block services and .NET device software to enable a new generation of smart Internet devices.
- Microsoft .NET products and services
Includes Microsoft® Windows.NET (with a core integrated set of building block services), MSN.NET, personal subscription services, Microsoft® Office.NET, Microsoft® Visual Studio.NET, and Microsoft® bCentral™ for .NET.
- Third-party .NET services
A vast range of partners and developers will have the opportunity to produce corporate and vertical services built on the .NET platform.

What is the .NET Framework?

The Microsoft .NET Framework is a platform for building, deploying, and running Web Services and applications. It provides a highly productive, standards-based, multi-language environment for integrating existing investments with next-generation applications and services as well as the agility to solve the challenges of deployment and operation of Internet-scale applications. The .NET Framework consists of three main parts: the common language runtime, a hierarchical set of unified class libraries, and a componentized version of Active Server Pages called ASP.NET. (Please see this paper for more information)

Does the .NET Framework only apply to people building Web sites?

The .NET Framework enables creating great Web applications. However, the .NET Framework also enables building the same applications you build today. If you write Windows software (using ATL/COM, MFC, Microsoft® Visual Basic®, or even standard Microsoft® Win32®), .NET offers many advantages to the way you currently build applications. Of course, if you do develop Web sites, then the .NET Framework has a lot to interest you—starting with ASP.NET.

How can I get the .NET Framework Beta?

You can download the .NET Framework from the Web at:

<http://msdn.microsoft.com/net> or order both Visual Studio .NET and the .NET Framework from the Web for a minimal charge. The cost for the beta

will cover materials and shipping. For North American delivery addresses please go to:

<http://msdn.microsoft.com/vstudio/nextgen/beta.asp>

What are the minimum system requirements for .NET Framework Beta 2?

The hardware requirements for this Beta release will be higher, and its performance may be lower than the final .NET Framework release. Beta 2 version of .NET Framework has been tested on Windows 2000 (Server and Professional), Windows NT 4 (Server and Workstation), Windows Millennium Edition, Windows 98, Windows XP, and Windows .NET Server. Web Application development and some server side components require Windows NT 4 or Windows 2000. We recommend the following hardware configurations:

Client

- Processor: Intel Pentium class 90 MHz or higher
- RAM: 32 MB (96 MB or higher recommended)
- Video: 800x600, 256 colors

Server

- Processor: Intel Pentium class 133 MHz or higher
- RAM: 128 MB (256 MB or higher recommended)
- Hard disk space required to install: 360 MB
- Hard disk space required: 210 MB
- Additional space required to install and compile all samples: 300 MB
- Video: 800x600, 256 colors

What are the major changes to the .NET Framework between Beta 1 and Beta 2 versions?

Top Eight Changes to .NET Framework

Change	Explanation
Win95 support discontinued	With most customers running Win 98/Me, Windows NT 4.0, and Windows 2000, it made sense to concentrate development and test resources on these platforms to ensure the reliability customers demand from Microsoft development tools.

Namespace changes	After extensive usability studies, Microsoft has tuned the namespaces to be more intuitive, therefore increasing developer productivity. For a complete list of all namespace changes, see APIchangesbeta1tobeta2.htm with the .NET Framework SDK.
Support for XML Schema Document (XSD)	W3C is endorsing XML Schema as a replacement for DTDs and the .NET Framework enables developers to use this system of managing XML documents.
Managed Win32 Classes removed	The .NET Framework Unified Classes provide developers with all the functionality needed to write XML Web services and applications, in an easier, intuitive, hierarchical way. If there is a need to call into unmanaged dll's this too is fully supported through the PInvoke() function.

Runtime Security Enhancements	
Policy and Permission enhancements	New policy level for enterprise policy enterprise policy administration. Execution permission checking is now ON by default.
Configuration and Administration features	New XML schema for security policy configuration. GUI admin tool contains security admin support (MSCORCFG.exe). Tools contain enterprise policy deployment support
Isolated Storage	Now supports a Roaming Store
Advances in Cryptography	<ul style="list-style-type: none"> Support for the Rijndael/AES block cipher and the SHA256, SHA384 and SHA512 hash algorithms. New streaming model for wrapping cryptographic transforms (e.g. symmetric block ciphers) around any Stream class. Support for RFC 3075 XML Digital Signatures
Granular resource protection	Permissions now enable a more granular protection of a number of resources including SQLServer drivers, OLEDB providers, the event log, perf counters, the active directory, printing, MSMQ, and Services.
Windows Forms	Enables code access security so that a very rich subset of the Windows Forms functionality can be used by semi-trusted code

ASP.NET Security Enhancements

Support for code access security	Administrators can assign permissions to web applications and services at the application level.
Configuration for FormsAuthentication	Forms authentication now supports configuration files for reading cookie values and encryption keys for a given application.
Control over process identity	ASP.NET enables administrators to dynamically set the identity under which the application runs

ASP.NET not supported on Windows NT 4.0

Focusing testing and development for Windows 2000 and beyond will ensure the best possible reliability(and performance) going forward

Configuration and Administration Tools

New GUI administration tools that uses XML schema for setting deployment policies and other administration configuration settings particularly in areas of security.

These Tools Include:

- .NET Framework Configuration (Mscorcfg.msc)
- .NET Framework Configuration Wizards (Mscorcfg.exe)

For a complete list of all of the changes from Beta 1 to Beta 2 of the .NET Framework see **APIchangesbeta1tobeta2.htm** in the .NET Framework SDK.

Where is Visual InterDev®?

The Visual Interdev Web development system has been included into the Visual Studio .NET IDE. All languages will have access to the market leading features of the current Visual InterDev product.

When will the .NET Framework be available?

Firm dates have not yet been set for final availability. As with all Microsoft products, customers let us know when a product is ready to release – the .NET Framework is no exception. Check back on <http://msdn.Microsoft.com/net> for more information.

Can my Visual Basic 6.0-based applications be upgraded to Visual Basic.NET?

In order to take full advantage of the new language constructs in Visual Basic.NET, an upgrade tool is included that automatically upgrades Visual

Basic 6 projects when they are opened in Visual Basic.NET. The Upgrade Tool will modify the language for syntax changes and will convert Visual Basic 6 forms to Windows Forms. In addition, to help developers upgrade their Visual Basic 6 projects, the Upgrade Tool generates a report which alerts developers to manual changes that need to be made in their code. Because these comments are displayed as "TO DO" tasks in the new Task List window, developers can easily navigate to the code statement simply by double-clicking the task.

How does Microsoft Visual C++ support .NET?

Microsoft Visual C++ can produce code for the .NET platform. In addition we have extensively improved ATL, MFC and the compiler. We have also introduced ATL Server for producing native code XML Web service application.

Where can I learn more about Microsoft Visual Studio .NET and how to build apps with it?

You can get more information from <http://msdn.microsoft.com/vstudio>. Additionally MSDN will be carrying a number of articles and white papers on .NET. Finally there are the newsgroups on MSNEWS.MICROSOFT.COM.

How does .NET support my existing COM and COM+ based components?

The .NET Framework enables developers to call existing COM and COM+ components as well as enabling .NET components to be exposed to COM and COM+ developers. Additionally a considerable amount of COM+ 1.0 “plumbing,” has been simplified in the .NET Framework. Additionally .NET Framework components can be added to a COM+ application where they can take advantage of the automatic Component Services – e.g., Transactions, ObjectPool, Queued Components, Events, etc , making building serviced components even easier.

Will there be a cross platform version of the runtime?

There are currently no plans to take the full .NET Framework to non-Windows platforms. However, there is a version of the .NET Framework called the .NET Compact Framework that enables the fundamental architecture of the .NET Framework on small devices such as mobile phones and digital TVs. The .NET Compact Framework is programming language, hardware, and operating system independent.

How do I secure my XML Web services?

Since XML Web services look and act externally like Web pages (using HTTP requests and responding with text which happens to be well formed XML), they are protected in much the same way that Web sites are protected: through standard web server authentication mechanisms and through the use of HTTPS and SSL to protect the data on the network (or IPSec and/or firewalls to further restrict access to client machines of known IP addresses).

Can I use the Win32[®] API from a .NET Framework program?

Yes. Using P/Invoke, .NET Framework-based applications can access native code libraries via static DLL entry points.

What programming languages will the .NET Framework support?

The .NET Framework is language neutral; virtually any language can target the .NET Framework. Currently, you can build .NET programs in a number of languages, including C++, Microsoft[®] Visual Basic.NET, JScript[®], and Microsoft's newest language—C#. A large number of third-party languages will also be available for building .NET Framework applications. These languages include COBOL, Eiffel, Perl, Python, Smalltalk, and others. For a complete list please see: www.gotdotnet.com

APPENDIX B

.NET FRAMEWORK TOOLS

The Microsoft .NET Framework SDK tools are designed to make it easier for you to create, deploy, and manage applications and components that target the .NET Framework. This section contains detailed information about the tools that ship with the .NET Framework SDK.

Configuration and Deployment Tools

Tool	Description
Assembly Cache Viewer (Shfusion.dll)	Allows you to view and manipulate the contents of the global assembly cache using Windows Explorer.
Assembly Linker (Al.exe)	Generates a file with an assembly manifest from one or more files that are either resource files or Microsoft intermediate language (MSIL) files.
Assembly Registration Tool (Regasm.exe)	Reads the metadata within an assembly and adds the necessary entries to the registry, which allows COM clients to create .NET Framework classes transparently.
Assembly Binding Log Viewer (Fuslogvw.exe)	Displays details for failed assembly binds. This information helps you diagnose why the .NET Framework cannot locate an assembly at runtime.
Global Assembly Cache Tool (Gacutil.exe)	Allows you to view and manipulate the contents of the global assembly cache. While Shfusion.dll provides similar functionality, you can use Gacutil.exe from build scripts, makefile files, and batch files.
Installer Tool (Installutil.exe)	Allows you to install and uninstall server resources by executing the installer components of a specified assembly.
Isolated Storage Tool (Storeadm.exe)	Lists or removes all existing stores for the currently logged-on user.
Native Image Generator (Ngen.exe)	Creates a native image from a managed assembly and installs it into the native image cache on the local computer.
.NET Framework Configuration Tool (Mscorcfg.msc)	Provides a graphical interface for managing .NET Framework security policy and applications that use remoting services. This tool also allows you to manage and configure assemblies in the global assembly cache.
.NET Services Installation Tool (Regsvcs.exe)	Adds managed classes to Windows 2000 Component Services by loading and registering the assembly and generating, registering, and installing the type library into an existing COM+ 1.0 application.
Soapsuds Tool (Soapsuds.exe)	Helps you compile client applications that communicate with XML Web services by using a technique called remoting.
Type Library Exporter (Tlbexp.exe)	Generates a type library from a common language runtime assembly.
Type Library Importer (Tlbimp.exe)	Converts the type definitions found within a COM type library into equivalent definitions in managed metadata format.

Web Services Description Language Tool (Wsdll.exe)	Generates code for ASP.NET XML Web services and XML Web services clients from Web Services Description Language (WSDL) contract files, XML Schema Definition (XSD) schema files, and .discomap discovery documents.
Web Services Discovery Tool (Disco.exe)	Discovers the URLs of XML Web services located on a Web server, and saves documents related to each XML Web service on a local disk.
XML Schema Definition Tool (Xsd.exe)	Generates XML schemas that follow the XSD language proposed by the World Wide Web Consortium (W3C). This tool generates common language runtime classes and DataSet classes from an XSD schema file.

Debugging Tools

Tool	Description
Microsoft CLR Debugger (DbgCLR.exe)	Provides debugging services with a graphical interface to help application developers find and fix bugs in programs that target the runtime.
Runtime Debugger (Cordbg.exe)	Provides command-line debugging services using the common language runtime Debug API. Used to find and fix bugs in programs that target the runtime.

Security Tools

Tool	Description
Certificate Creation Tool (Makecert.exe)	Generates X.509 certificates for testing purposes only.
Certificate Manager Tool (Certmgr.exe)	Manages certificates, certificate trust lists (CTLs), and certificate revocation lists (CRLs).
Certificate Verification Tool (Chktrust.exe)	Verifies the validity of a file signed with an X.509 certificate.
Code Access Security Policy Tool (Caspol.exe)	Allows you to examine and modify machine, user, and enterprise-level code access security policies.
File Signing Tool (Signcode.exe)	Signs a portable executable (PE) file with an Authenticode digital signature.
Permissions View Tool (Permview.exe)	Displays the minimal, optional, and refused permission sets requested by an assembly. You can also use this tool to view all declarative security used by an assembly.
Secutil Tool (Secutil.exe)	Extracts strong name public key information or Authenticode publisher certificates from an assembly, in a format that can be incorporated into code.
Set Registry Tool (Setreg.exe)	Allows you to change the registry settings for the Software Publishing State keys, which control the behavior of the certificate verification process.
Software Publisher Certificate Test Tool (Cert2spc.exe)	Creates, for test purposes only, a Software Publisher's Certificate (SPC) from one or more X.509 certificates.

Strong Name Tool (Sn.exe)	Helps create assemblies with strong names. Sn.exe provides options for key management, signature generation, and signature verification.
---------------------------	--

General Tools

Tool	Description
Common Language Runtime Minidump Tool (Mscordmp.exe)	Creates a file containing information that is useful for analyzing system issues in the runtime. The Microsoft Dr. Watson tool (Drwatson.exe) invokes this program automatically.
License Compiler (Lc.exe)	Reads text files that contain licensing information and produces a .licenses file that can be embedded in a common language runtime executable.
Management Strongly Typed Class Generator (Mgmtclassgen.exe)	Allows you to quickly generate an early-bound class in C#, Visual Basic, or JScript for a specified Windows Management Instrumentation (WMI) class.
MSIL Assembler (Ilasm.exe)	Generates a PE file from Microsoft intermediate language (MSIL). You can run the resulting executable, which contains MSIL code and the required metadata, to determine whether the MSIL code performs as expected.
MSIL Disassembler (Ildasm.exe)	Takes a PE file that contains MSIL code and creates a text file suitable as input to the MSIL Assembler (Ilasm.exe).
PEVerify Tool (PEverify.exe)	Performs MSIL type safety verification checks and metadata validation checks on a specified assembly.
Resource File Generator Tool (Resgen.exe)	Converts text files and .resx (XML-based resource format) files to .NET common language runtime binary .resources files that can be embedded in a runtime binary executable or compiled into satellite assemblies.
Windows Forms ActiveX Control Importer (Aximp.exe)	Converts type definitions in a COM type library for an ActiveX control into a Windows Forms control.
Windows Forms Class Viewer (Wincv.exe)	Finds managed classes matching a specified search pattern, and displays information about those classes using the Reflection API.
Windows Forms Resource Editor (Winres.exe)	Allows you to quickly and easily localize Windows Forms forms.

APPENDIX C

GLOSSARY

Term	Definition
.NET Framework	The .NET Framework is a platform for building the next generation of distributed XML Web services and applications. It exposes a language-independent yet consistent programming model across all tiers of an application, while providing seamless interoperability with and easy migration from existing technologies. The .NET Framework consists of three things: the common language runtime, unified classes, and ASP.NET.
ADO.NET	Data Access technology for the .NET Framework.
ASP.NET	Active Server Pages technology for the .NET Framework
Assembly	The unit of deployment and versioning in the .NET Framework. It establishes the namespace for resolving requests and determines which resources are exposed externally and which are accessible from within the assembly. An assembly includes an Assembly manifest which describes the contents of the assembly
C#	The first component orient language in the C/C++ family. Submitted to ECMA for standardization.
CGI	Common Gateway Interface – early Internet protocol used to generate interactive content over the Web.
Common Language Infrastructure (CLI)	A subset of the .NET Framework submitted to ECMA for Standardization
Common Language Runtime (CLR)	The type, metadata and execution systems provided by the .NET Framework, which supplies managed code and data with services such as cross-language integration, code access security, object lifetime management, and debugging and profiling support. By targeting the CLR, compilers and other tools can offer these services to developers
Common Language Specification (CLS)	A subset of the .NET Framework's features that are supported by a broad set of compliant languages and tools. CLS-compliant languages and tools are guaranteed to interoperate with other CLS-compliant languages and tools. For example, the type <code>Int32</code> is CLS compliant and languages and tools may expect that other CLS languages and tools know how to correctly use this type.
Disco	SOAP Discovery (Disco) - based on SOAP Discovery specification, provides a set of rules for discovery of Web services description and capabilities.
ECMA	A European standards body created in 1961. Internationally accredited ECMA has fast track approval for ISO and is the forum for successful standards like ECMAScript.
Evidence-based security	The .NET Framework introduces the concept of <i>evidence-based</i> security, referring to inputs to the security policy about code such as from what site, security zone or URL was an assembly obtained, what is its strong name, and whether it has a digital signature and from whom. Based upon these and other answers, the appropriate security policy can be applied, and the appropriate permissions may be granted to the assembly. Answers can come from multiple sources, including one or more of the CLR, the browser, ASP.NET, and the shell, depending on the source of the code.
Garbage collection	The process of transitively tracing through all pointers to actively used objects to locate all objects that can be referenced and then arranging to reuse any heap memory that was not found during this trace. The CLR's garbage collector also arranges to compact the memory that is in use to reduce the working space needed for the heap.

HTTP	Hyper Text Transfer Protocol, standard Internet protocol for transfer of information between client/servers and server/server...
IDL	Interface Definition Language – a language used by application to specify the various interfaces they intend to offer to other applications.
Intermediate Language (IL)	Intermediate Language (IL) is a language used as the output of a number of compilers and as the input to a JIT compiler. IL defines an abstract stack-based execution architecture. The CLR may include several JIT compilers for converting IL to native code.
JIT	Just-In-Time - a phrase that describes an action that is taken only when it becomes necessary, such as Just-In-Time compilation or Just-In-Time object activation. By convention, the term "JIT" alone is used to refer to a JIT compiler.
Loosely coupled application	A distributed application that you can change the implementation of one tier without affecting any of the other tiers. Contrast tightly coupled architecture.
Managed code	Managed code supplies the metadata necessary for the CLR to provide services such as memory management, cross-language integration, code access security, and automatic lifetime control of objects. All code based on IL executes as managed code.
Manifest	Metadata describing which modules and resource files are part of a particular assembly, which types are exported and which other assemblies are referenced. It also specifies which security permissions are required to run, what additional permissions are optionally requested, and what permissions the assembly refuses
Metadata	Data (or information) about data. In the CLR, metadata is used to describe assemblies and types. It is stored with them in the executable files, and is used by compilers, tools, and the runtime to provide a wide range of services. Metadata is essential for runtime type information and dynamic method invocation. Many systems use metadata, for example Type Libraries in COM provide metadata
Native code	Code that has been compiled to processor-specific machine code.
N-tier	System architecture that separates presentation, business logic, data access, and database (or other persistence mechanism) tiers.
Reflection	.NET Framework technology that allows you to examine metadata that describes types and their members.
SOAP	Simple Object Access Protocol, WC3 standard - a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses.
Tightly coupled architecture	A distributed application where a change to any tier affects some or all the other remaining tiers. Contrast loosely coupled architecture.
UDDI	The Universal Description, Discovery and Integration [UDDI] specification – an initiative that creates a global, platform-independent, open framework to enable businesses to (1) discover each other, (2) define how they interact over the Internet, and (3) share information in a global registry
Unified classes	The .NET Framework's unified, object-oriented, hierarchical, and extensible set of class libraries ("APIs") that developers can use from the languages they are already familiar with.

Unmanaged code	Code that was created without knowledge for the conventions and requirements of .NET Framework. Unmanaged code executes in the .NET Framework environment with minimal services (for example, no garbage collection, limited debugging , no declarative security). Unmanaged code does not have metadata describing it.
Web API	APIs that enable integration of an XML Web service under the .NET Framework.
Web Forms	Web Forms are an ASP.NET technology that you use to create programmable Web pages. They can present information, using any markup language, to the user in any browser and use code on the server to implement application logic.
WebMethod	Programming keyword in .NET Framework that enables objects to be Internet accessible.
Windows Forms	The Windows Forms framework encapsulates native Win32 APIs and exposes secure, managed classes for creating Win32 client-side applications. The Windows Forms class library provides many controls such as buttons, check boxes, drop-down lists, combo boxes, data grid, and others that encapsulate user interface and other client-side functionality.
WSDL	Web Service Description Language – an XML grammar that developers and development tools use to represent the capabilities of an XML Web service
XML	Extensible Markup Language – WC3 standard for the format of structured documents and data on the Web.
XML Web service	An XML Web service is an application that exposes its functionality programmatically over the Internet or intranet using standard Internet protocols and standards such as HTTP and XML.

For More Information

For the latest information on the .NET Framework, please see
<http://msdn.microsoft.com/net>

