.NET Framework

Web Application and Web Services

Contents

Introduction

This document describes the .NET Framework and the salient features it provides to develop Web Applications and Web Services. It also provides an introduction to latest version of the .NET Framework - .NET Framework 3.0.

.NET Framework

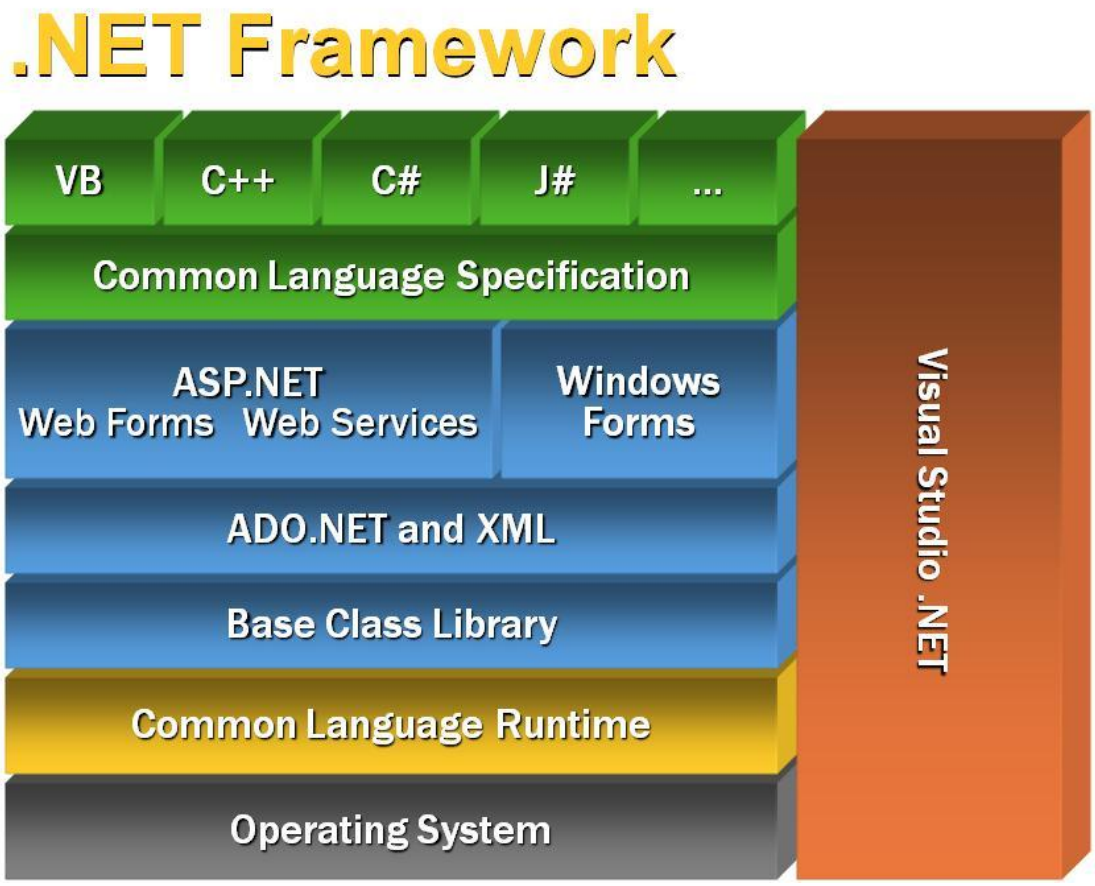
The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library.

.NET Framework has evolved around for more than five years now as a fully released platform with different versions though it has been available as Beta even earlier than that. The current version of .NET Framework is 3.0 with support for Windows Presentation Foundation, Windows Communication, Windows Workflow Foundation and Windows Card Space. The base class libraries provide an object oriented programming model for developing Web based Applications as well as Services unlike the traditional scripting model.

The object oriented model provides more flexibility in programming and better recycling of components, libraries as well as making them secured and scalable.



The above block diagram represents the various parts of the .NET Framework and how they run on top of one common platform which provides the base class library, the data access components and a language independent programming model.

There are various integral components of the .NET Framework and they can elaborately be identified into the following:-

Common Language Runtime (CLR)

The Common Language Runtime is the runtime engine that executes .NET Framework Applications. In the case of .NET Framework Web Applications, the .NET Framework needs to be installed on the Web Server hosting the application and in the case of Windows based Desktop Applications, .NET Framework gets installed in the individual system.

The CLR helps in Just In Time Compilation of the MSIL Code that is generated by the .NET Compiler. The MSIL is language independent and it is generated from a high level language like C#.

The virtual machine aspect of the CLR allows programmers to ignore many details of the specific CPU that will execute the program. The CLR also provides other important services, including the following:

- Memory management
- Thread management

- Exception handling
- Garbage collection
- Security

Base Class Library

The .NET Framework base class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object-oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

The .NET Framework types are the foundation on which .NET applications, components, and controls are built. The .NET Framework includes types that perform the following functions:

- Represent base data types and exceptions.
- Encapsulate data structures.
- Perform I/O.
- Access information about loaded types.
- Invoke .NET Framework security checks.
- Provide data access, rich client-side GUI, and server-controlled, client-side GUI.

The .NET Framework provides a rich set of interfaces, as well as abstract and concrete (non-abstract) classes. You can use the concrete classes as is or, in many cases, derive your own classes from them. To use the functionality of an interface, you can either create a class that implements the interface or derive a class from one of the .NET Framework classes that implements the interface.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

- Console applications.
- Windows GUI applications (Windows Forms).
- ASP.NET applications.
- XML Web services.
- Windows services.

Common Type System

The common type system defines how types are declared, used, and managed in the runtime, and is also an important part of the runtime's support for cross-language integration. The common type system performs the following functions:

- Establishes a framework that helps enable cross-language integration, type safety, and high performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.
- Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.

Metadata and Self-describing Components

In the past, a software component (.exe or .dll) written in one language could not easily use a software component written in another language. COM provided a step forward in solving this problem. The .NET Framework makes component interoperation even easier by allowing compilers to emit additional declarative information into all modules and assemblies. This information, called metadata, helps components to seamlessly interact.

Cross-Language Interoperability

The common language runtime provides built-in support for language interoperability. However, this support does not guarantee that code you write can be used by developers using another programming language. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them, called the Common Language Specification (CLS), has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

Assemblies

Assemblies are the building blocks of .NET Framework applications; they form the fundamental unit of deployment, version control, reuse, activation scoping, and security permissions. An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality. An assembly provides the common language runtime with the information it needs to be aware of type implementations. To the runtime, a type does not exist outside the context of an assembly.

Application Domains

Operating systems and runtime environments typically provide some form of isolation between applications. For example, Microsoft Windows uses processes to isolate applications. This isolation is

necessary to ensure that code running in one application cannot adversely affect other, unrelated applications.

Application domains provide an isolation boundary for security, reliability, and versioning, and for unloading assemblies. Application domains are typically created by runtime hosts, which are responsible for bootstrapping the common language runtime before an application is run.

Runtime Hosts

The common language runtime has been designed to support a variety of different types of applications, from Web server applications to applications with a traditional rich Windows user interface. Each type of application requires a runtime host to start it. The runtime host loads the runtime into a process, creates the application domains within the process, and loads user code into the application domains.

The .NET Framework ships with a number of different runtime hosts, including the hosts listed in the following table.

Runtime Host	Description
ASP.NET	Loads the runtime into the process that is to handle the Web request. ASP.NET also creates an application domain for each Web application that will run on a Web server.
Microsoft Internet Explorer	Creates application domains in which to run managed controls. The .NET Framework supports the download and execution of browser-based controls. The runtime interfaces with the extensibility mechanism of Microsoft Internet Explorer through a mime filter to create application domains in which to run the managed controls. By default, one application domain is created for each Web site.
Shell executables	Invokes runtime hosting code to transfer control to the runtime each time an executable is launched from the shell.

Microsoft provides a set of APIs for writing your own runtime hosts.

See also Overview of .NET Framework

ASP.NET

ASP.NET combines unprecedented developer productivity with performance, reliability, and deployment.

Developer Productivity

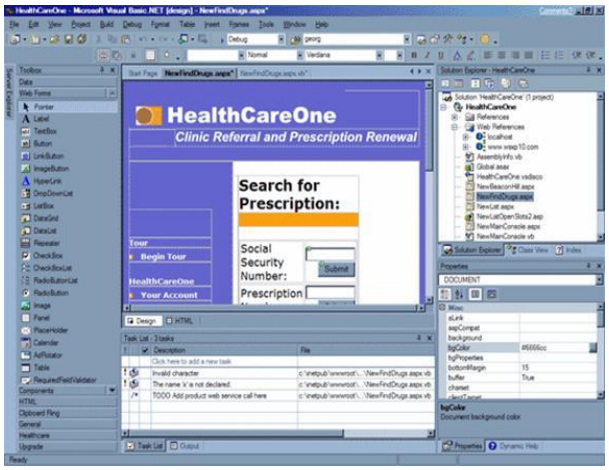
ASP.NET helps you deliver real world Web applications in record time.

Easy Programming Model

ASP.NET makes building real world Web applications dramatically easier. ASP.NET server controls enable an HTML-like style of declarative programming that let you build great pages with far less code than with classic ASP. Displaying data, validating user input, and uploading files are all amazingly easy. Best of all, ASP.NET pages work in all browsers -- including Netscape, Opera, AOL, and Internet Explorer.

Flexible Language Options

ASP.NET lets you leverage your current programming language skills. Unlike classic ASP, which supports only interpreted VBScript and JScript, ASP.NET now supports more than 25 .NET languages (including built-in support for VB.NET, C#, and JScript.NET -- no tool required), giving you unprecedented flexibility in your choice of language.



Great Tool Support

You can harness the full power of ASP.NET using any text editor -- even Notepad! But Visual Studio 2005 adds the productivity of Visual Basic-style development to the Web. Now you can visually design ASP.NET Web Forms using familiar drag-drop-double click techniques, and enjoy full-fledged code support including statement completion and color-coding. VS.NET also provides integrated support for debugging and deploying ASP.NET Web applications.

The Professional version of Visual Studio 2005 delivers life-cycle features to help organizations plan, analyze, design, build, test, and coordinate teams that develop ASP.NET Web applications. These include UML class modeling, database modeling (conceptual, logical, and physical models), testing tools (functional, performance and scalability), and enterprise frameworks and templates, all available within the integrated Visual Studio .NET environment.

Rich Class Framework

Application features that used to be hard to implement, or required a 3rd-party component, can now be added in just a few lines of code using the .NET Framework. The .NET Framework offers over 4500 classes that encapsulate rich functionality like XML, data access, file upload, regular expressions, image generation, performance monitoring and logging, transactions, message queuing, SMTP mail, and much more!

Improved Performance and Scalability

ASP.NET lets you use serve more users with the same hardware.

Compiled execution

ASP.NET is much faster than classic ASP, while preserving the "just hit save" update model of ASP. However, no explicit compile step is required! ASP.NET will automatically detect any changes, dynamically compile the files if needed, and store the compiled results to reuse for subsequent requests. Dynamic compilation ensures that your application is always up to date, and compiled execution makes it fast. Most applications migrated from classic ASP see a 3x to 5x increase in pages served.

Rich output caching

ASP.NET output caching can dramatically improve the performance and scalability of your application. When output caching is enabled on a page, ASP.NET executes the page just once, and saves the result in memory in addition to sending it to the user. When another user requests the same page, ASP.NET serves the cached result from memory without re-executing the page. Output caching is configurable, and can be used to cache individual regions or an entire page. Output caching can dramatically improve the performance of data-driven pages by eliminating the need to query the database on every request.

Web-Farm Session State

ASP.NET session state lets you share session data user-specific state values across all machines in your Web farm. Now a user can hit different servers in the web farm over multiple requests and still have full access to her session. And since business components created with the .NET Framework are free-threaded, you no longer need to worry about thread affinity.

Microsoft .NET Outperforms J2EE

In a head-to-head comparison of performance and scalability between Sun's Java Pet Store J2EE blueprint application and the ASP.NET implementation, Microsoft .NET significantly outperformed

J2EE. The bottom line: the ASP.NET implementation required only 1/4th as many lines of code, was 28x faster (that's 2700%), and supported 7.6x as many concurrent users as J2EE, with only 1/6th as much processor utilization.

Enhanced Reliability

ASP.NET ensures that your application is always available to your users.

Memory Leak, DeadLock and Crash Protection

ASP.NET automatically detects and recovers from errors like deadlocks and memory leaks to ensure your application is always available to your users.

For example, say that your application has a small memory leak, and that after a week the leak has tied up a significant percentage of your server's virtual memory. ASP.NET will detect this condition, automatically start up another copy of the ASP.NET worker process, and direct all new requests to the new process. Once the old process has finished processing its pending requests, it is gracefully disposed and the leaked memory is released. Automatically, without administrator intervention or any interruption of service, ASP.NET has recovered from the error.

Easy Deployment

ASP.NET takes the pain out of deploying server applications.

"No touch" application deployment

ASP.NET dramatically simplifies installation of your application. With ASP.NET, you can deploy an entire application as easily as an HTML page: just copy it to the server. No need to run regsvr32 to register any components, and configuration settings are stored in an XML file within the application.

Dynamic update of running application

ASP.NET now lets you update compiled components without restarting the web server. In the past with classic COM components, the developer would have to restart the web server each time he deployed an update. With ASP.NET, you simply copy the component over the existing DLL -- ASP.NET will automatically detect the change and start using the new code.

Easy Migration Path

You don't have to migrate your existing applications to start using ASP.NET. ASP.NET runs on IIS side-by-side with classic ASP on Windows 2000 and Windows XP platforms. Your existing ASP applications continue to be processed by ASP.DLL, while new ASP.NET pages are processed by the new ASP.NET engine. You can migrate application by application, or single pages. And ASP.NET even lets you continue to use your existing classic COM business components.

New Application Models

ASP.NET extend your application's reach to new customers and partners.

XML Web Services

XML Web services allow applications to communicate and share data over the Internet, regardless of operating system or programming language. ASP.NET makes exposing and calling XML Web Services simple.

Any class can be converted into an XML Web Service with just a few lines of code, and can be called by any SOAP client.

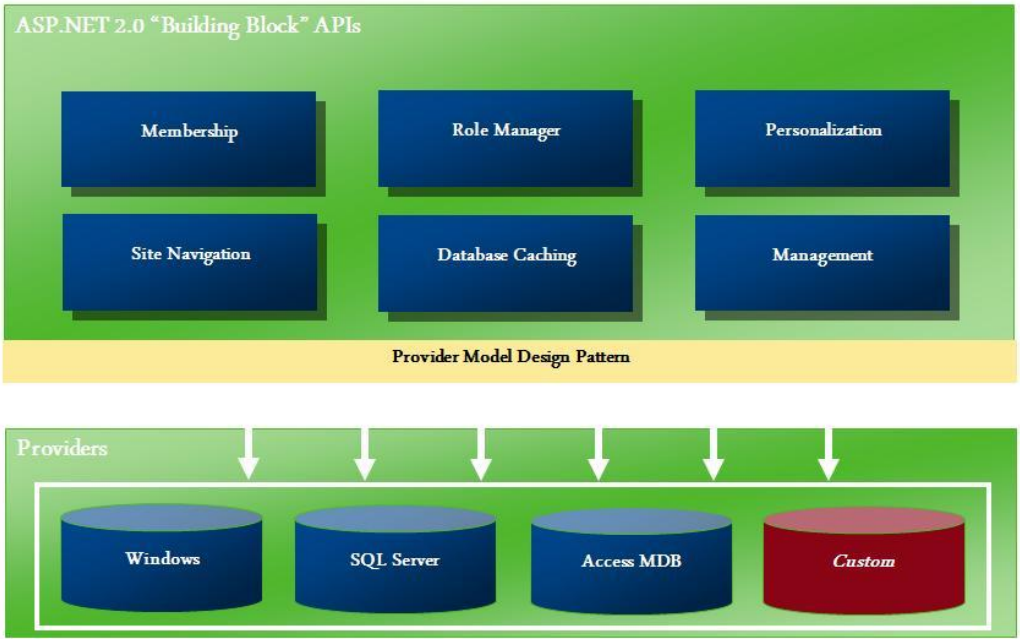
Likewise, ASP.NET makes it incredibly easy to call XML Web Services from your application. No knowledge of networking, XML, or SOAP is required.

Mobile Web Device Support

ASP.NET Mobile Controls let you easily target cell phones, PDAs -- over 80 mobile Web devices -- using ASP.NET. You write your application just once, and the mobile controls automatically generate WAP/WML, HTML, or iMode as required by the requesting device.

Also See ASP.NET Developer Center

ASP.NET 2.0 Developer Stack



ASP.NET is the Web Development Model for developing web applications and Services which target the .NET Framework. ASP.NET is a robust, declarative programming model for developing web based applications as well as XML Web Services.

ASP.NET 2.0, the latest version provides various utilities which are common in development scenarios in the form of built-in features, controls which make web development a very quick and efficient model. ASP.NET 2.0 runs on top of .NET Framework 2.0 and runs well with Internet Information Services (IIS) and Windows Server Systems.

ASP.NET 2.0 has various advantages over other web development models. We will explore some of the features as below:-

- Declarative Programming
- Code-Behind Model (which separates Design from Code)
- Built-in Data Controls such as DataGridView, DataGrid
- Built-in mechanisms such as Caching, Paging, Tracing and Profiling
- Built-in advanced features such as Personalization, Membership Management
- Easy to Deploy and Maintain

ADO.NET

ADO.NET is a data-access technology that enables applications to connect to data stores and manipulate data contained in them in various ways. It is based on the .NET Framework and it is highly integrated with the rest of the Framework class library.

The ADO.NET API is designed so it can be used from all programming languages that target the .NET Framework, such as Visual Basic, C#, J# and Visual C++.

The ADO.NET stack has two major parts: providers and services.

ADO.NET "providers" are the components that know how to talk to specific data stores (for example, there is a provider to talk to SQL Server databases, and another one to talk to Oracle databases). All providers surface a unified API on top of which other layers can be built.

ADO.NET also includes services built on top of the providers that are designed to facilitate writing applications. One such service is support for an in-memory cache that retains the relational shape of the data, and does change-tracking and constraint validation among other things; this service surfaces through the ADO.NET DataSet, and includes components to interact with the provider layer.

ADO.NET is part of the .NET Framework, so in order to use ADO.NET in your applications all you need is the .NET Framework installed in the computers where your application will be used. The .NET Framework can be downloaded from the .NET Framework Developer Center.

Databases for ADO.NET

SQL Server 2005

SQL Server 2005 is a comprehensive database platform providing enterprise-class data management with integrated business intelligence (BI) tools. The SQL Server 2005 database engine provides more secure, reliable storage for both relational and structured data, enabling you to build and manage highly available, performant data applications for use in your business.

ADO.NET includes first-class support for SQL Server 2005 right out of the box. So you don't need to install any other software in addition to Visual Studio and SQL Server in order to get started.

Other Databases

ADO.NET also includes support for other databases through ADO.NET providers. The .NET Framework includes providers for SQL Server and Oracle databases; it also includes bridge providers that allow you to use your legacy OLEDB and ODBC drivers.

Additionally, there are various companies that build ADO.NET providers for many other databases.

See also ADO.NET, .NET Data Access Architecture Guide

.NET Compact Framework

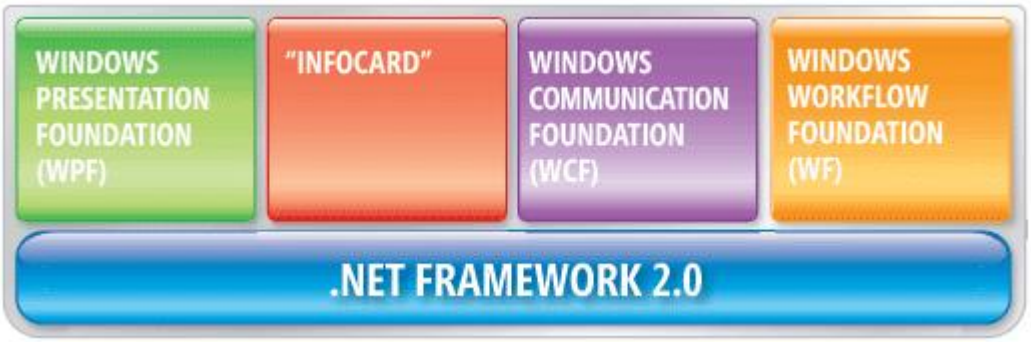
The Microsoft .NET Compact Framework is a key part of realizing Microsoft's goal to provide customers with great experiences any time, any place, and on any device. The .NET Compact Framework's managed code and XML Web services enable the development of secure, downloadable applications on devices such as personal digital assistants (PDAs), mobile phones, and set-top boxes.

The Microsoft .NET Compact Framework is the smart device development framework for Microsoft .NET, bringing the world of managed code and XML Web services to devices. The .NET Compact Framework is a rich subset of the .NET Framework, thus providing the same benefits as the .NET Framework. But the .NET Compact Framework is designed specifically for resource-constrained devices, such as PDAs and smart mobile phones. The .NET Compact Framework greatly simplifies the process of creating and deploying applications to mobile devices while also allowing the developer to take full advantage of the capabilities of the device.

See Also .NET Compact Framework

.NET Framework 3.0

The Microsoft .NET Framework 3.0 (formerly WinFX), is the new managed code programming model for Windows. It combines the power of the .NET Framework 2.0 with new technologies for building applications that have visually compelling user experiences, seamless communication across technology boundaries, and the ability to support a wide range of business processes.



Windows Presentation Foundation

The Microsoft Windows Presentation Foundation provides the foundation for building applications and high fidelity experiences in Windows Vista, blending together application UI, documents, and media content, while exploiting the full power of your computer. The functionality extends to the support for Tablet and other forms of input, a more modern imaging and printing pipeline, accessibility and UI automation infrastructure, data driven UI and visualization, as well as the integration points for weaving the application experience into the Windows shell.

Windows Communication Foundation

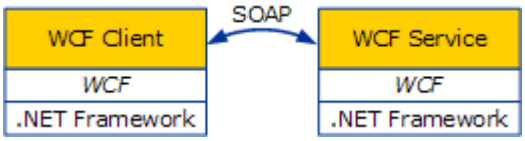
The global acceptance of Web services, which includes standard protocols for application-to-application communication, has changed software development. For example, the functions that

Web services now provide include security, distributed transaction coordination, and reliable communication. The benefits of the changes in Web services should be reflected in the tools and technologies that developers use. Windows Communication Foundation (WCF) is designed to offer a manageable approach to distributed computing, broad interoperability, and direct support for service orientation.

WCF simplifies development of connected applications through a new service-oriented programming model. WCF supports many styles of distributed application development by providing a layered architecture. At its base, the WCF channel architecture provides asynchronous, untyped message-passing primitives. Built on top of this base are protocol facilities for secure, reliable, transacted data exchange and broad choice of transport and encoding options.

The typed programming model (called the *service model*) is designed to ease the development of distributed applications and to provide developers with expertise in ASP.NET Web Services, .NET remoting, and Enterprise Services and who are coming to WCF with a familiar development experience. The service model features a straightforward mapping of Web services concepts to those of the .NET Framework Common Language Runtime (CLR), including flexible and extensible mapping of messages to service implementations in languages such as Visual C# or Visual Basic. It includes serialization facilities that enable loose coupling and versioning, and it provides integration and interoperability with existing .NET distributed systems technologies such as MSMQ, COM+, ASP.NET Web Services, Web Services Enhancements (WSE), and a number of other functions.

The foundation for new Windows-based applications is the .NET Framework. Accordingly, WCF is implemented primarily as a set of classes on top of the .NET Framework CLR. Because it extends their familiar environment, WCF enables developers who create object-oriented applications using the .NET Framework today to also build service-oriented applications in a familiar way.



The figure shows a view of a WCF client and service. The two interact using SOAP, WCF's native message representation, so even though the figure shows both parties built on WCF, this is not required. WCF is built on .NET Framework 2.0.

WCF addresses a range of challenges for communicating applications. Three things stand out, however, as WCF's most important aspects:

- Unification of existing .NET Framework communication technologies.
- Support for cross-vendor interoperability, including reliability, security, and transactions.
- Explicit service orientation.

Unification of Microsoft Distributed Computing Technologies

In the absence of WCF, the development team that implements the distributed application would need to choose the right distributed technology from the multiple choices offered by the .NET Framework. Yet given the diverse requirements of the distributed application, no single technology

would fit the requirements. Instead, the application would probably use multiple existing .NET Framework technologies, such as the following:

- **ASMX (also called ASP.NET Web Services)**. An option for communicating with the J2EE-based existing application and with the partner applications across the Internet. Given that basic Web services are supported today on most platforms, this was the most direct way to achieve cross-vendor interoperability before the release of WCF.

- .**NET Remoting**. An option for communication with the other .NET application, because both are built on the .NET Framework. Remoting is designed expressly for tightly coupled .NET-to-.NET communication, so it offers a seamless and straightforward development experience for applications in the local network.

- **Enterprise Services**. Used by the application for managing object lifetimes and defining distributed transactions. These functions could be useful in communicating and integrating with any of the other applications, but Enterprise Services supports only a limited set of communication options.

- **Web Services Enhancements (WSE)**. Could be used along with ASMX to communicate with the J2EE-based application and with the partner applications. Because it implements more recently defined Web services agreements, known collectively as the WS-* specifications, WSE allows for more flexible Web services security, as long as all applications involved support compatible versions of these new specifications.

- **Microsoft Message Queuing (MSMQ)**. Used to communicate with Windows-based applications that require guaranteed data delivery as well as decoupling of workloads and application lifetimes. The durable messaging that Message Queuing provides is typically the best solution for intermittently connected applications.

Built on .NET Framework, the distributed application must use more than one of these communication technologies to meet its requirements. Although this is technically possible, the resulting application would be complex to implement and challenging to maintain.

With WCF, the solution is much easier to implement. As the figure shows, WCF can be used for all the situations previously described. Accordingly, the distributed application can use this single technology for all of its application-to-application communication. The following shows how WCF addresses each of these requirements:

- Because WCF can communicate using Web services, interoperability with other platforms that also support SOAP, such as the leading J2EE-based application servers, is straightforward.

- WCF can also be configured and extended to communicate with Web services using messages not based on SOAP, for example, simple XML formats like RSS.

- To allow optimal performance when both parties in a communication are built on WCF, the wire encoding used in this case is an optimized binary version of an XML Information Set. Messages still conform to the data structure of a SOAP message, but their encoding uses a

binary representation of that data structure rather than the standard angle-brackets-and-text format of the XML 1.0 text encoding. Using this option makes sense for communicating with the call center client application, because it is also built on WCF, and performance is an important concern.

- Managing object lifetimes, defining distributed transactions, and other aspects of Enterprise Services are now provided by WCF. They are available to any WCF-based application, which means that the rental car reservation application can use them with any of the other applications it communicates with.
- Because it supports a large set of the WS-* specifications, WCF helps provide reliability, security, and transactions when communicating with any platform that also supports these specifications.
- WCF's option for queued messaging, built on Message Queuing, allows applications to use persistent queuing without using another set of application programming interfaces.

The result of this unification is greater functionality and significantly reduced complexity.

Interoperability with Applications Built on Other Technologies

While WCF introduces a new development environment for distributed applications, it is designed to interoperate well with the non-WCF applications. There are two important aspects to WCF interoperability: interoperability with other platforms, and interoperability with the Microsoft technologies that preceded WCF. The following section describes both.

Interoperability with Other Web Services Platforms

Enterprises today typically have systems and applications that they purchased from a range of suppliers. In most enterprise distributed applications, for instance, communication is required with various other software applications written in various languages and running on various operating systems.

Because WCF's fundamental communication mechanism is SOAP-based Web services, WCF-based applications can communicate with other software running in a variety of contexts. An application built on WCF can interact with all of the following:

- WCF-based applications running in a different process on the same Windows machine.
- WCF-based applications running on another Windows machine.
- Applications built on other technologies, such as J2EE application servers, that support standard Web services. These applications can be running on Windows machines or on machines running other operating systems.

To allow more than just basic communication, WCF implements Web services technologies defined by the WS-* specifications. All of these specifications were originally defined by Microsoft, IBM, and other vendors working together. As the specifications become stable, ownership often passes to standards bodies, such as the World Wide Web Consortium (W3C) or the Organization for the Advancement of Structured Information Standards (OASIS). These specifications address several areas, including basic messaging, security, reliability, transactions, and working with a service's

metadata. For more information, see Interoperability and Integration. For more information about advanced Web services specifications, see http://msdn.microsoft.com/webservices/webservices/understanding/default.aspx.

Grouped by function, those specifications cover:

- Messaging: SOAP is the foundation for Web services and defines a basic envelope that contains header and a body sections. WS-Addressing defines additions to the SOAP header for addressing SOAP messages, which frees SOAP from relying on the underlying transport protocol, such as HTTP, to carry addressing information. Message Transmission Optimization Mechanism (MTOM) defines an optimized transmission format for SOAP messages with large binary data contents based on the XML-binary Optimized Packaging (XOP) specification.

- Metadata: The Web Services Description Language (WSDL) defines a standard language for specifying services and various aspects of how those services can be used. WS-Policy allows specification of more dynamic aspects of a service's behavior that cannot be expressed in WSDL, such as a preferred security option. WS-MetadataExchange allows a client to directly request descriptive information about a service, such as its WSDL and its policies, using SOAP.

- Security: WS-Security, WS-SecureConversation, WS-Trust, and WS-Federation all define additions to SOAP messages for providing authentication, data integrity, data privacy, and other security features.

- Reliability: WS-Reliable Messaging defines additions to the SOAP header that allow reliable end-to-end communication, even when one or more Web services intermediaries must be traversed.

- Transactions: Built on WS-Coordination, WS-Atomic Transaction allows coordinating two-phase commit transactions in the context of Web services conversations.

The distributed applications would likely use several of these more advanced technologies. For example, WS-Addressing is essential whenever SOAP is used over a transport mechanism other than HTTP, which might be the case for communication with the .NET Framework-based call center client application. WCF relies on WS-Policy and WS-Metadata Exchange to discover whether the system it is communicating with is also using WCF and for other things. Reliable communication is essential for most situations, so it is likely that WS-Reliable Messaging would be used to interact with many of the other applications in this scenario. Similarly, WS-Security and the related specifications might also be used for securing the communication with one or more of the applications, because all would require some kind of protection against unauthorized access or message modification and interception. For the applications that require transaction integration with the rental car reservation system, WS-Atomic Transaction would be essential. Finally, MTOM could be used whenever an optimized wire format for binary data is necessary (for instance for pictures of fleet examples), and both sides of the communication supported this option.

The key point is that WCF implements interoperable Web services complete with cross-platform security, reliability, transactions, and other services. To provide maximum throughput, WCF-to-WCF

communication can be significantly optimized, but all other communication uses standard Web services protocols. In fact, it is possible for a single application to expose its services to both kinds of clients.

Interoperability with Microsoft Technologies

Many Microsoft customers have made significant investments in the .NET Framework technologies that WCF includes. Protecting those investments was a fundamental goal of WCF's designers. Installing WCF does not break existing technology, so there is no requirement that organizations change existing applications to use it. A clear upgrade path is provided, however, and wherever possible, WCF interoperates with those earlier technologies.

For example, both WCF and ASMX use SOAP, so WCF-based applications can directly interoperate with those built on ASMX. Existing Enterprise Services applications can also be wrapped with WCF interfaces, allowing them to interoperate with applications built on WCF. And because WCF's persistent queuing relies on MSMQ, WCF-based applications can interoperate directly with non-WCF-based applications built using native MSMQ interfaces. In the rental car reservations application, software built using any of these earlier technologies could directly connect to and use the new system's WCF-based services.

Interoperability is not always possible, however. For example, even though WSE 1.0 and WSE 2.0 implement some of the same WS-* specifications as WCF, these earlier technologies implement earlier versions of the specifications. Version 3.0 of WSE does allow interoperability with WCF, but earlier versions do not. For more information about interoperability, see Migrating WSE 3.0 Web Services to WCF.

Interoperability with Other XML Protocols

The future of the Internet is not predictable. It is not predictable and the technologies used today may evolve or be replaced. For example, a popular trend in building web-centric applications (called by many "Web 2.0"), is an application model based on communication using only simple XML formats that are not SOAP-based and exclusively relying on HTTP as a transport as well as application protocol. Trying to extend the model of the interactive Web to all kinds of distributed applications, the Representational State Transfer (REST) architectural style has no notion of user-defined operations for dealing with data, but the application state is rather immediately associated with HTTP URLs and the intrinsic HTTP methods (PUT, POST, DELETE, GET) are immediately used to convey commands. This approach is contrast to the notion of user-defined procedures or functions that most developers are used to in an enterprise environment, but is of value in scenarios where services are designed to function as the back end of Web 2.0 applications. In this evolving environment of community-driven conventions, experimental programming models, ongoing reinterpretation and refinement of standards, flexibility is required to cope with unforeseeable changes. WCF is prepared for such changes. While it uses the SOAP information model as an underlying structure, it is not bound to using SOAP for wire commuinication. In fact, WCF can be configured to support "plain" XML and accept and emit XML data without any SOAP envelope wrapper. WCF can also be extended to support specific XML formats, such as the IETF content syndication-feed standard ATOM (and the popular RSS community convention), and even non-XML formats, such as JavaScript Object Notation (JSON). This flexibility ensures that code written today will be valid in the future, even if protocols change or are replaced.

See Also WCF, Overview of WCF

Windows Card Space ("Info Card")

The Internet continues to be increasingly valuable, and yet also faces significant challenges. Online identity theft, fraud, and privacy concerns are rising. Users must track a growing number of accounts and passwords. This burden results in "password fatigue," and that results in insecure practices, such as reusing the same account names and passwords at many sites. Many of these problems are rooted in the lack of a widely adopted identity solution for the Internet.

Windows CardSpace is Microsoft's implementation of an Identity Metasystem that enables users to choose from a portfolio of identities that belong to them and use them in contexts where they are accepted, independent of the underlying identity systems where the identities originate and are used.

Opportunities and Challenges

For users and businesses alike, the Internet continues to be increasingly valuable. More people are using the Web for everyday tasks, from shopping, banking, and paying bills to consuming media and entertainment. E-commerce is growing, with businesses delivering more services and content across the Internet, communicating and collaborating online, and inventing new ways to connect with each other.

But as the value of what people do online has increased, the Internet itself has become more complex and dangerous. Online identity theft, fraud, and privacy concerns are on the rise. And increasingly sophisticated practices such as "phishing" are invented. In response, a multitude of systems designed to protect identity have been devised. The diversity results in the aforementioned password fatigue and unsafe practices.

The root of these problems is that the Internet was designed without a system of digital identity in mind. In efforts to address this deficiency, numerous digital identity systems have been introduced, each with its own strengths and weaknesses. But no single system meets the requirements of every digital identity scenario. The reality is that many different identity systems are in use today, with still more being invented. The result is an inconsistent patchwork of improvised solutions at every Web site, rendering the system as a whole fragile, and constraining the fuller realization of the promise of e-commerce.

Open Identity Metasystem

Given that universal adoption of a single digital identity system or technology is unlikely ever to occur, a successful and widely employed identity solution for the Internet requires a different approach—one with the capability to connect existing and future identity systems into an identity metasystem (or "system of systems"). This metasystem leverages the strengths of its constituent identity systems, provides interoperability between them, and enables creation of a consistent and straightforward user interface to all. The resulting improvements in cyberspace benefit everyone, making the Internet a safer place with the potential to boost e-commerce, combat phishing, and solve other digital identity challenges.

Maintain the Diversity of Systems

In the offline world, people carry multiple forms of identification in their wallets, such as driver's licenses or other government-issued identity cards, credit cards, and cards such as frequent flyer cards. People control which card to use and how much information to reveal in any given situation.

Similarly, the identity metasystem makes it easier for users to stay safe and in control when accessing resources on the Internet. It lets users select an identity from among a portfolio of their digital identities and use them at Internet services of their choice where they are accepted. The metasystem enables identities provided by one identity system technology to be used within systems based on different technologies, provided an intermediary exists that understands both technologies and is willing and trusted to do the required translations.

It is important to note that the identity metasystem does not compete with or replace the identity systems it connects. Instead, the goals of the identity metasystem are to connect individual identity systems, allowing seamless interoperation between them, to provide applications with a technology-independent representation of identities, and to provide a better, more consistent user experience with all of them. The metasystem relies on the individual systems to do its work.

Identities in Context

The identities held by a person in the offline world can range from the significant, such as birth certificates, passports, and drivers' licenses, to the trivial, such as business cards or frequent coffee buyer's cards. People use their different forms of identification in different contexts where they are accepted.

Identities can be in or out of context. Identities used out of context generally do not bring the desired result. For example, trying to use a coffee card to cross a border is clearly out of context. On the other hand, using a bank card at an ATM, a government-issued ID at a border, a coffee card at a coffee stand, and a Passport Network (formerly .NET Passport) account at MSN Hotmail are all clearly in context.

In some cases, the distinction is less clear. You can use a government-issued ID at your ATM instead of a bank-issued card, but if this resulted in the government having knowledge of each financial transaction, some people would be uncomfortable. You can use a Social Security Number as a student ID number, but that facilitates identity theft. And you can use Passport accounts at some non-Microsoft sites, but few sites chose to enable this; even where it was enabled, few users did so because they felt that Microsoft's participation in these interactions was out of context.

Studying the Passport experience and other digital identity initiatives throughout the industry led to working with a wide range of industry experts to derive a set of principles that are fundamental to a successful, broadly adopted, and enduring digital identity system on the Internet. The following section describes these principles.

Principles ("Laws of Identity")

The open identity metasystem is designed to follow a set of principles (also called "The Laws of Identity") that have been developed with ongoing feedback and input from a broad community of people active in the digital identity community.

The principles that an identity system should follow are the following.

User Control and Consent

Identity systems reveal information that identifies a user only with the user's consent.

Minimal Disclosure for a Constrained Time

The identity system solution that discloses the least amount of identifying information is the most stable, long-term solution.

Justifiable Parties

Identity systems disclose identifying information only to parties who have a necessary and justifiable place in a given identity relationship.

Directed Identity

Identity systems support both "omnidirectional" identifiers for use by public entities and "unidirectional" identifiers for use by private entities, thus facilitating discovery while preventing unnecessary release of correlation handles.

Pluralism of Operators and Technologies

Identity systems channel and enable the inner workings of multiple identity technologies run by multiple identity providers.

Human Integration

Identity systems define the human user to be a component of the distributed system, integrated through unambiguous human-machine communications mechanisms that offer protection against identity attacks.

Consistent Experience Across Contexts

Identity systems facilitate negotiation between a relying party and user of a specific identity. That presents a harmonious human and technical interface while permitting the autonomy of identity in different contexts.

Open Identity Metasystem Architecture

This section covers the general architecture of an open identity metasystem.

Roles

Different parties participate in the metasystem in different ways. The following roles within the metasystem are:

Identity Providers issue identities. For example, credit-card providers might issue identities that enable payment, businesses might issue identities to their customers, governments might issue identities to citizens, and individuals might use self-issued identities in contexts like signing on to Web sites.

Relying Parties require identities. For example, a Web site or online service that uses identities offered by other parties.

Subjects are the individuals and other entities about who claims are made. Examples include end users, companies, and organizations.

Each person and entity that participates in an identity metasystem can play all the roles, and each person and entity can play more than one role at a time. Often a person or entity plays all three roles simultaneously.

Components

This section describes the key components and concepts of the metasystem.

The metasystem is made up of five key components:

- A way to represent identities using claims.
- A means for identity providers, relying parties, and subjects to negotiate.
- An encapsulating protocol to obtain claims and requirements.
- A means to bridge technology and organizational boundaries using claims transformation.
- A consistent user experience across multiple contexts, technologies, and operators.

Claims-Based Identities

Identities consist of sets of claims that are asserted about the subject of the identity. For example, the claims on a driver's license might include the issuing state, the driver's license number, a name, address, gender, birth date, the kinds of vehicles the licensee is eligible to drive, and so on. The issuing state asserts that these claims are valid.

The claims on a credit card might include the card issuer's identity, the card-holder's name, the account number, the expiration date, the validation code, and the card-holder's signature. The card issuer asserts that these claims are valid.

The claims on a self-issued identity (such as a business card) might include your name, address, and telephone number. For self-issued identities, you assert that these claims are valid yourself.

Negotiation

Negotiation enables participants in the metasystem to make agreements required for them to connect with one another within the metasystem. Negotiation is used to determine mutually acceptable technologies, claims, and requirements. For instance, if one party understands SAML and X.509 claims, and another understands Kerberos and X.509 claims, the parties negotiate and decide to use X.509 claims with one another. Another type of negotiation determines whether the claims required by a relying party can be supplied by a particular identity. Both kinds of negotiation are simple matching exercises; they compare what one party can provide with what the other one requires to determine whether there is a fit.

Encapsulating Protocol

The encapsulating protocol provides a technology-neutral way to exchange claims and requirements between subjects, identity providers, and relying parties. The participants determine the content and meaning of what is exchanged, not the metasystem. For example, the encapsulating protocol would allow an application to retrieve SAML-encoded claims without having to understand or implement the SAML protocol.

Claims Transformers

Claims transformers bridge organizational and technical boundaries by translating claims understood in one system into claims understood and trusted by another system, thereby insulating the mass of

clients and servers from the intricacies of claim evaluation. Claims transformers may also transform or refine the semantics of claims. For example, a claim asserting, "Is an employee" might be transformed into the new claim, "OK to purchase book." The claim "Born on March 22, 1960" could be transformed into the claim "Age is over 21 years," which intentionally supplies less information. Claims transformers may also be used to change claim formats. For instance, claims made in formats such as X.509, Kerberos, SAML 1.0, SAML 2.0, SXIP, and others could be transformed into claims expressed using different technologies. Claims transformers provide the interoperability needed today, plus the flexibility required to incorporate new technologies.

Consistent User Experience

Many identity attacks succeed because the user was fooled by something presented on the screen, not because of insecure communication technologies. For example, phishing attacks occur not in the secured channel between Web servers and browsers—a channel that might extend thousands of miles—but in the two or three feet between the browser and the human who uses it. The identity metasystem, therefore, seeks to empower users to make informed and reasonable identity decisions by enabling the development of a consistent, comprehensible, and integrated user interface for making those choices.

One key to securing the whole system is to present an easy-to-learn, predictable user interface that looks and works the same no matter which underlying identity technologies are employed. Another key is making important information obvious—for instance, displaying the identity of the site you are authenticating to in a way that makes spoofing attempts apparent. The user must be informed which items of personal information relying parties are requesting, and for what purposes. This allows users to make informed choices about whether or not to disclose this information. Finally, the user interface provides a means for the user to actively consent to the disclosure, if they agree to the conditions.

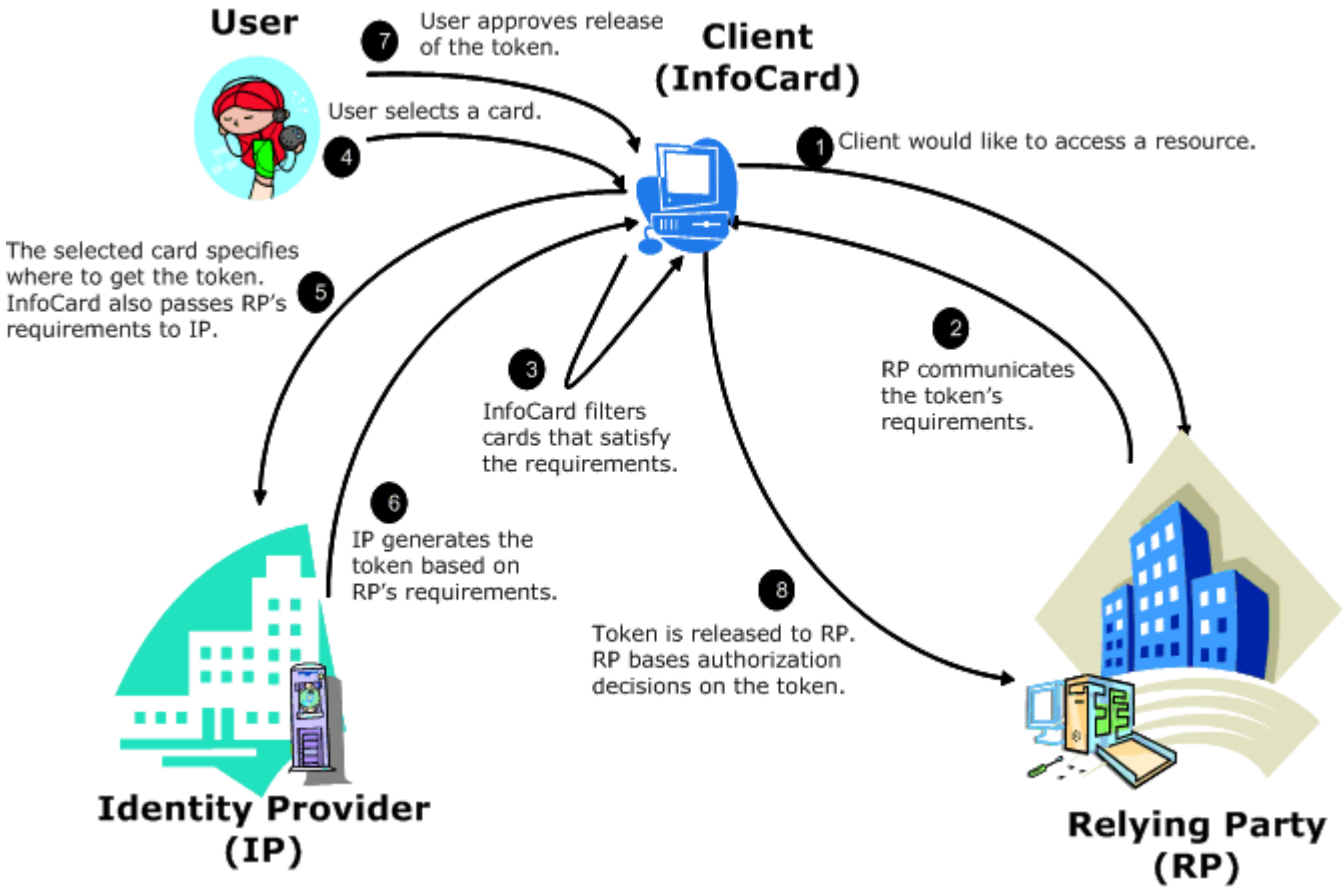
WS- Specifications*

As with other features of WCF, the Windows CardSpace technology is built upon a set of open specifications, the WS-* Web Services Architecture. The encapsulating protocol used for claims transformation is WS-Trust. Negotiations are conducted using WS-MetadataExchange and WS-SecurityPolicy. These protocols enable building a technology-neutral identity metasystem and form the "backplane" of the identity metasystem. Like other Web services protocols, they also allow new kinds of identities and technologies to be incorporated and used as they are developed and adopted by the industry.

To foster the interoperability necessary for broad adoption, the specifications for WS-* are published and are freely available, have been and continue to be submitted to open standards bodies, and allow implementations to be developed royalty-free.

End-to-End Scenario

The following figure illustrates the end-to-end processes that occur when you use Windows CardSpace to access a site that requires user validation.



Many of the problems on the Internet today, from phishing attacks to inconsistent user experiences, come from the patchwork nature of digital identity solutions that software makers have built in the absence of a unifying and architected system of digital identity. An identity metasystem, as defined by the Laws of Identity, supplies a unifying fabric of digital identity, uses existing and future identity systems, provides interoperability between them, and enables the creation of a consistent and straightforward user interface to them all. Basing our efforts on the Laws of Identity, Microsoft is working with others in the industry to build the identity metasystem using published WS-* protocols that render Microsoft's implementations fully interoperable with those produced by others. Microsoft's implementation of components of the identity metasystem is the Windows CardSpace system.

Using Windows CardSpace, many of the dangers, complications, annoyances, and uncertainties of today's online experiences can be a thing of the past. Widespread deployment of the identity metasystem has the potential to solve many of these issues, benefiting everyone and accelerating the long-term growth of connectivity by making the online world safer, more trustworthy, and easier to use. Microsoft is working with others in the industry to define and deploy the identity metasystem.

See Also Using CardSpace in Windows Communication Foundation

Windows Workflow Foundation

Windows Workflow Foundation is the programming model, engine, and tools for quickly building workflow-enabled applications on Windows. It consists of a namespace, an in-process workflow

engine, and designers for Visual Studio 2005. Windows Workflow Foundation can be developed and run on Windows Vista, Windows XP, and the Windows Server 2003 family. Windows Workflow Foundation includes support for both system workflow and human workflow across a variety of scenarios, including workflow within line-of-business applications, user interface page-flow, document-centric workflow, human workflow, composite workflow for service-oriented applications, business rule-driven workflow, and workflow for systems management.

Windows Workflow Foundation provides a consistent and familiar development experience with other .NET Framework 3.0 technologies, such as Windows Communication Foundation and Windows Presentation Foundation. It provides full support for Visual Basic .NET and C#, debugging, a graphical workflow designer, and developing your workflow completely in code or in markup. Windows Workflow Foundation also provides an extensible model and designer to build custom activities that encapsulate workflow functionality for end users or for reuse across multiple projects.

See Also Windows Workflow Foundation.

.NET Web Services

An XML Web service is a programmable entity that provides a particular element of functionality, such as application logic, and is accessible to any number of potentially disparate systems using ubiquitous Internet standards, such as XML and HTTP. XML Web services depend heavily upon the broad acceptance of XML and other Internet standards to create an infrastructure that supports application interoperability at a level that solves many of the problems that previously hindered such attempts.

An XML Web service can be used internally by a single application or exposed externally over the Internet for use by any number of applications. Because it is accessible through a standard interface, an XML Web service allows heterogeneous systems to work together as a single web of computation.

Instead of pursuing the generic capabilities of code portability, XML Web services provide a viable solution for enabling data and system interoperability. XML Web services use XML-based messaging as a fundamental means of data communication to help bridge the differences that exist between systems that use incongruent component models, operating systems, and programming languages. Developers can create applications that weave together XML Web services from a variety of sources in much the same way that developers traditionally use components when creating a distributed application.

One of the core characteristics of an XML Web service is the high degree of abstraction that exists between the implementation and the consumption of a service. By using XML-based messaging as the mechanism by which the service is created and accessed, both the XML Web service client and the XML Web service provider are freed from needing any knowledge of each other beyond inputs, outputs, and location.

XML Web services are enabling a new era of distributed application development. It is no longer a matter of object model wars or programming language beauty contests. When systems are tightly coupled using proprietary infrastructures, this is done at the expense of application interoperability. XML Web services deliver interoperability on an entirely new level that negates such

counterproductive rivalries. As the next revolutionary advancement of the Internet, XML Web services will become the fundamental structure that links together all computing devices.

See Also Web Services and the .NET Framework

Security

About .NET Security

The common language runtime and the .NET Framework provide many useful classes and services that enable developers to easily write security code. These classes and services also enable system administrators to customize the access that code has to protected resources. In addition, the runtime and the .NET Framework provide useful classes and services that facilitate the use of cryptography and role-based security.

Code Access Security

Code access security is a mechanism that helps limit the access that code has to protected resources and operations. In the .NET Framework, code access security performs the following functions:

- Defines permissions and permission sets that represent the right to access various system resources.
- Enables administrators to configure security policy by associating sets of permissions with groups of code (code groups).
- Enables code to request the permissions it requires in order to run, as well as the permissions that would be useful to have, and specifies which permissions the code must never have.
- Grants permissions to each assembly that is loaded, based on the permissions requested by the code and on the operations permitted by security policy.
- Enables code to demand that its callers have specific permissions.
- Enables code to demand that its callers possess a digital signature, thus allowing only callers from a particular organization or site to call the protected code.
- Enforces restrictions on code at run time by comparing the granted permissions of every caller on the call stack to the permissions that callers must have.

To determine whether code is authorized to access a resource or perform an operation, the runtime's security system walks the call stack, comparing the granted permissions of each caller to the permission being demanded. If any caller in the call stack does not have the demanded permission, a security exception is thrown and access is refused. The stack walk is designed to help prevent luring attacks, in which less-trusted code calls highly trusted code and uses it to perform unauthorized actions. Demanding permissions of all callers at run time affects performance, but it is essential to help protect code from luring attacks by less-trusted code. To optimize performance,

you can have your code perform fewer stack walks; however, you must be sure that you do not expose a security weakness whenever you do this.

See Also Code Access Security

Role-based security

Roles are often used in financial or business applications to enforce policy. For example, an application might impose limits on the size of the transaction being processed depending on whether the user making the request is a member of a specified role. Clerks might have authorization to process transactions that are less than a specified threshold, supervisors might have a higher limit, and vice-presidents might have a still higher limit (or no limit at all). Role-based security can also be used when an application requires multiple approvals to complete an action. Such a case might be a purchasing system in which any employee can generate a purchase request, but only a purchasing agent can convert that request into a purchase order that can be sent to a supplier.

.NET Framework role-based security supports authorization by making information about the principal, which is constructed from an associated identity, available to the current thread. The identity (and the principal it helps to define) can be either based on a Windows account or be a custom identity unrelated to a Windows account. .NET Framework applications can make authorization decisions based on the principal's identity or role membership, or both. A role is a named set of principals that have the same privileges with respect to security (such as a teller or a manager). A principal can be a member of one or more roles. Therefore, applications can use role membership to determine whether a principal is authorized to perform a requested action.

To provide ease of use and consistency with code access security, .NET Framework role-based security provides PrincipalPermission objects that enable the common language runtime to perform authorization in a way that is similar to code access security checks. The **PrincipalPermission** class represents the identity or role that the principal must match and is compatible with both declarative and imperative security checks. You can also access a principal's identity information directly and perform role and identity checks in your code when needed.

The .NET Framework provides role-based security support that is flexible and extensible enough to meet the needs of a wide spectrum of applications. You can choose to interoperate with existing authentication infrastructures, such as COM+ 1.0 Services, or to create a custom authentication system. Role-based security is particularly well-suited for use in ASP.NET Web applications, which are processed primarily on the server. However, .NET Framework role-based security can be used on either the client or the server.

See Also Role based Security

Cryptographic Services

Public networks such as the Internet do not provide a means of secure communication between entities. Communication over such networks is susceptible to being read or even modified by unauthorized third parties. In addition to file encryption and encryption on a local disk, cryptography helps you create a secure means of communication over otherwise insecure channels, providing data integrity and authentication.

The classes in the .NET Framework cryptography namespace manage many details of cryptography for you. Some are wrappers for the unmanaged Microsoft CryptoAPI, while others are purely managed implementations. You do not need to be an expert in cryptography to use these classes. When you create a new instance of one of the encryption algorithm classes, keys are auto-generated for ease of use, and default properties are as safe and secure as possible.

See Also Cryptographic Services

Security Policy Management

Security policy is the configurable set of rules that the common language runtime follows when determining the permissions to grant to code. The runtime examines identifiable characteristics of the code, such as the Web site or zone where the code originates, to determine the access that code can have to resources. During execution, the runtime ensures that code accesses only the resources that it has been granted permission to access.

Security policy defines several code groups and associates each of them with a set of permissions. Code groups categorize code by characteristics such as its publisher, digital signature, the URL from where it originates, and so on. After all evidence is considered, code is placed into code groups and the resulting permission grant is the total set of permissions associated with every code group that the code obtains membership in. Although the default security policy is suitable for most situations, administrators can modify or customize security policy to tailor it to the specific needs of their organizations. The runtime grants permissions to both assemblies and application domains based on security policy.

Secure Coding Guidelines

Evidence-based security policy and code access security provide very powerful, explicit mechanisms to implement security. Most application code can simply use the infrastructure implemented by the .NET Framework. In some cases, additional application-specific security is required, built either by extending the security system or by using new ad hoc methods.

Using the .NET Framework-enforced permissions, and other enforcement in your code, you should erect barriers to prevent malicious code from obtaining information that you do not want it to have or performing other undesirable actions. Additionally, you must strike a balance between security and usability in all the expected scenarios using trusted code.

See Also Secure Coding Guidelines

ACL Technology

The classes in the System.Security.AccessControl namespace allow you to programmatically create or modify discretionary access control lists (DACLs) and system access control lists (SACLs) for a number of protected resources such as files, folders, and so on. DACLs allow you to programmatically control access to protected resources, while SACLs allow you to programmatically control system auditing policies of protected resources. For example, you can use the DACL classes to make sure that only an administrator can read a file; you can use the SACL classes to make sure that all successful attempts to open the file are logged.

See Also ACL Technology Overview

Web application security

ASP.NET has been built with security in mind. ASP.NET leverages Microsoft's Internet Information Server (IIS) to provide strong support for common HTTP authentication schemes including support for Basic, Digest, NTLM, Kerberos, and SSL/TLS client certificates.ASP.NET also supports Microsoft Passport authentication and provides a convenient implementation of Forms-based (Cookie) authentication. Regardless of which authentication scheme is employed, developers get a consistent programming and authorization model.

References

http://msdn.microsoft.com/netframework

http://msdn.microsoft.com/asp.net

http://msdn.microsoft.com/security

http://www.netfx3.com

http://msdn2.microsoft.com/en-us/netframework/aa497273.aspx