Windows® DNA
"Building Windows Applications for the Internet Age"

For some time now, both small and large companies have been building robust applications for personal computers that continue to be ever more powerful and available at ever-lower costs. While these applications are being used by millions of users each day, new forces are having a profound effect on the way software developers build applications today and the platform in which they develop and deploy their application.

The increased presence of Internet technologies is enabling global sharing of information. Not only from small and large businesses, but individuals as well. The Internet has sparked a new creativity in many, resulting in many new businesses popping up overnight, running twenty-four hours a day, seven days a week. Competition and the increased pace of change are putting ever increasing demands for an application platform that enables application developers to build and rapidly deploy highly adaptive applications in order to gain strategic advantage.

It is possible to think of these new Internet applications needing to handle literally millions of users – a scale difficult to imagine a just a few short years ago. As a result, applications need to deal with user volumes of this scale, reliable to operate 24 hours a day and flexible to meet changing business needs. The application platform that underlies these types of applications must also provide a coherent application model along with a set of infrastructure and pre-built services for enabling development and management of these new applications.

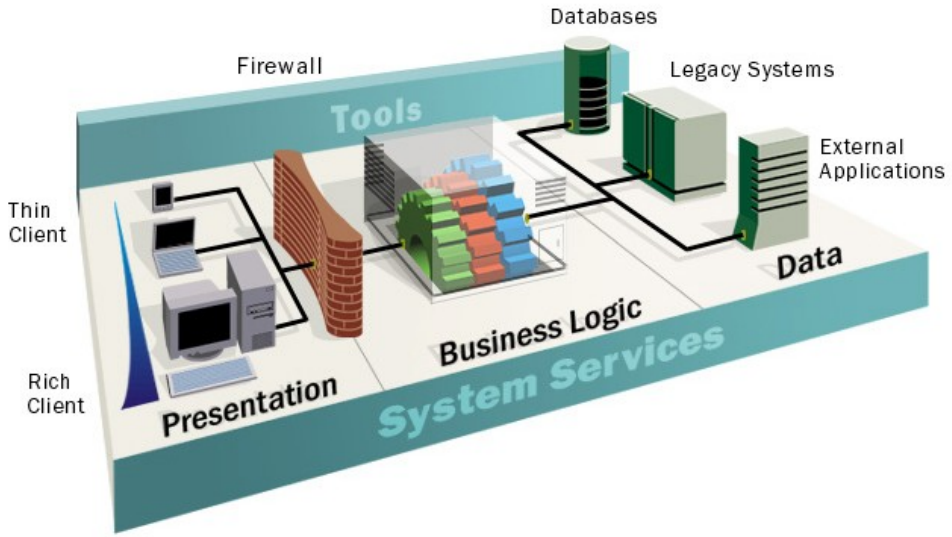

Figure 1. Windows DNA

The Microsoft® application platform consists of a multi-tiered distributed application model called Windows DNA (Figure 1) and a comprehensive set of infrastructure and application services. Windows DNA (http://www.microsoft.com/dna) unifies the best of the services available on personal computers, application servers and mainframes today; the benefits inherent in client-server computing and the best of Internet technologies around a common, component based application architecture. Simply put, guiding principles of Windows DNA are:

Internet Ready. Develop solutions that fully exploit the application platform's flexibility and the Internet's global reach and on-demand communication capabilities.

Faster Time to Market. Develop and deploy solutions rapidly without requiring developer reeducation or a paradigm shift in how software is built. Expose services and functionality through the underlying "plumbing" to reduce the amount of code developers must write.

True Interoperability. Build interoperability into all tiers so functionality can be added to existing systems. Adhere to open protocols and standards so other vendor solutions can be integrated.

Reduced Complexity. Integrate key services directly into the operating system and expose them in a unified way through the components. Reduce the need for IT professionals to function as system integrators so they can focus on solving the business problem.

Language, Tool and Hardware Independence. Provide a language-neutral component model so that developers can use task-appropriate tools. Build on the PC model of computing, wherein customers can deploy solutions on widely available hardware.

***Lower the Total Cost of Ownership*.** Develop applications that are easy to deploy, manage and change over time.

Given the proper underlying infrastructure, the multi-tier model of presentation, business logic and data can physically distribute processing over many computers. However, the core abstractions that have worked for single and two tier models in the past – high-level programming languages, database management systems, and graphical user interfaces – do not fully address the requirements of multi-tier application development. A different level of abstraction is needed to develop scalable, manageable and maintainable multi-user applications and at Microsoft, we believe this abstraction is cooperating components.

Cooperating Components

Microsoft's Windows DNA strategy rests on Microsoft's vision of cooperating components that are built based on the binary standard called the Component Object Model (COM). COM (http://www.microsoft.com/com) is the most widely used component software model in the world available on over 150 million desktop and servers today. It provides the richest set of integrated services, the widest choice of easy-to-use tools, and the largest set of available applications. In addition, it provides the only currently viable market for reusable, off-the-shelf, client and server components.

COM enables software developers to build applications from binary software components that can be deployed at any tier of the application model. These components provide support for packaging, partitioning, and distributed application functionality. COM enables applications to be developed with components by encapsulating any type of code or application functionality, such as a user interface control or line of business object. A component may have one or more interfaces; each exposes a set of methods and properties that can be queried and set by other components and applications. For example, a customer component might expose various properties such as name, address, and telephone number.

With the Microsoft Windows DNA model, components take away the complexity of building multi-tier applications. Applications based on components and the Windows DNA model rely on a common set of infrastructure and networking services provided in the Windows application platform. The Windows NT® security service, for example, provides access control to the Internet Information Server, as well as transaction and message queuing services. Other common services include systems management, directory services, networking, and hardware support.

Client Environments and Presentation Tier

Today many application developers using cooperating components target the development of their applications to the Windows platform to take full advantage of the rich user interface Windows has to offer. Likewise, customers have come to expect a rich, highly functional user interface from their applications. The extended reach of information and services to customers that the Internet has enabled has created a new challenge for the application developer. The application developer today must develop a user interface that is distributable, available on Windows and non-Windows platforms and supports a wide range of client environments, from handheld wireless devices to high-end workstations. Yet, applications must be rich with features to stay competitive and maintain the functionality that customers have come to expect.

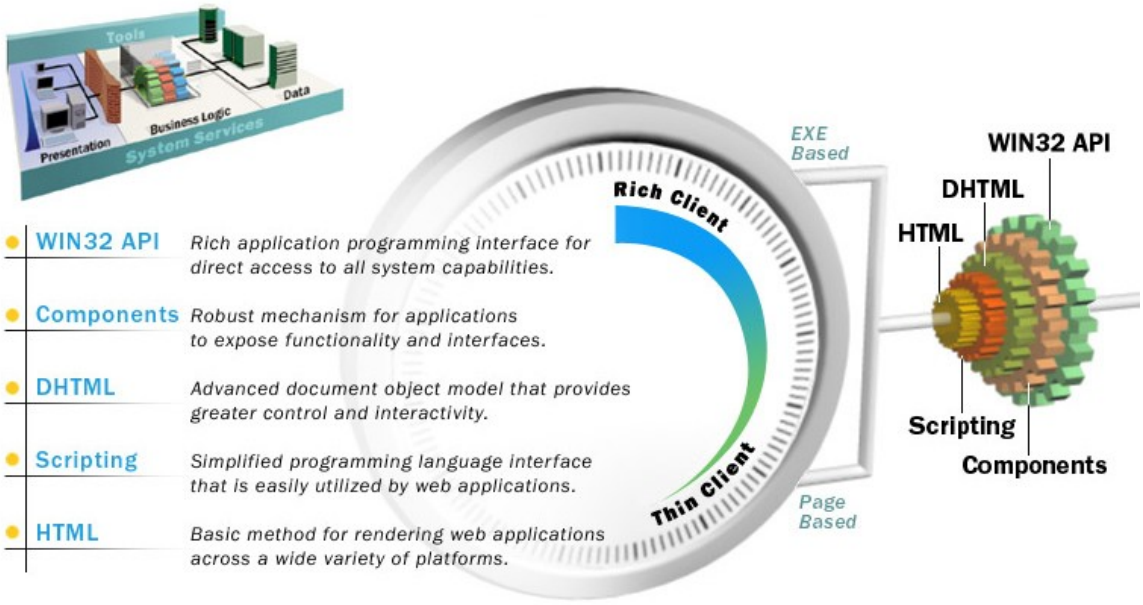

Figure 2. Windows DNA Presentation Services

As depicted in Figure 2, Windows DNA offers a broad range of presentation services giving the application developer the choice when developing the best solution. Windows DNA permits the developer to choose the appropriate Windows components and Internet technologies to support the richest possible interface and range of client environments, from handheld wireless devices to high-end workstations.

Maintaining broad reach to a wide range of client environments while achieving the greatest compatibility with all browsers, application developers will generally use standard HTML in developing their browser neutral applications. Microsoft tools and application services support the current generation of standard HTML.

The compromise in using static HTML is the reduced amount of functionality and richness in an applications user interface that customers have come to expect. This is OK for some applications as their application requires broad reach and browser neutrality.

There is a class of applications that don't have a browser neutrality requirement. The reality is that many corporations standardize on a single browser. In addition, application developers who want to provide more functionality in their application than they can achieve with standard HTML write code to determine the browser being used. These browser enhanced applications are written to take advantage of the technologies inherent in the browser to gain maximum

functionality and richness. With technologies like Dynamic HTML and Scripting, application developers can create actions with functional Web-based interfaces for data entry or reporting without using custom controls of applets.

DHTML is based on the W3C-standard Document Object Model, which makes all Web-page elements programmable objects. Think of DHTML like a “programmable” HTML. Contents of the HTML document, including style and positioning information can be modified dynamically by script code embedded in the page. Thus scripts can change the style, content and structure of a Web page without having to refresh the web page from the web server. By doing so, the client does not have to repeatedly return to the web server for changes in display resulting in increased network performance. Unlike Java applets or ActiveX® controls, DHTML has no dependencies on the underlying virtual machine or operating system. For clients without DHTML support, the content appears in a gracefully degraded form.

There are times when DHTML plus Scripting is not enough. Segments of applications need to leverage the operating system and underlying machine in which it is hosted on, while still maintaining an active connection to the Internet for data or additional services. It is in those instances in which application developers can take advantage of the components and Internet services provided by Windows to build more robust applications. Unlike Page-Based applications that are being run within the context of a browser, these applications are full fledged Windows Executables that have full access to the broad range of services provided by the Windows client. These applications generally use a combination of HTML, DHTML, Scripting, and ActiveX controls to provide rich integration with the client system as well as full connectivity to remote services on the Internet.

Applications written using the Win32® API offer the most functionality with reach limited to the application platforms that support the Win32 API. Through the use of cooperating components, developers can today have access to Internet technologies in the Windows application platform from within a Win32-based application. Some common examples are Microsoft Office 97 and Visual Studio® 98 development system. These applications support unified browsing, by embedding hyperlinks from within the application, host the browser for the display of documentation written in DHTML and provide the capability to download updates to the products over the Internet seamlessly.

Application Services

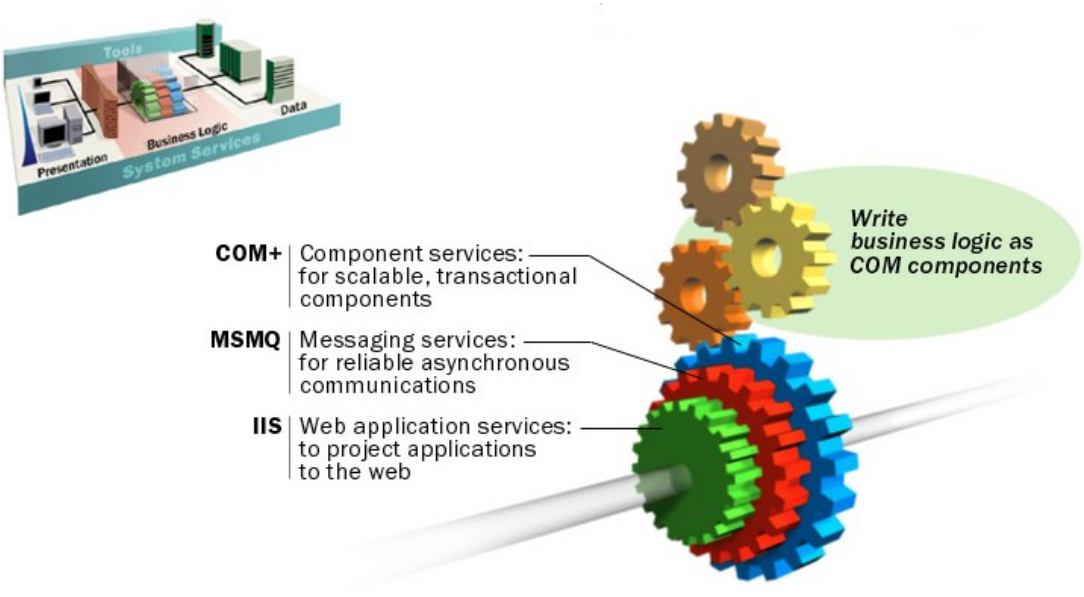

Figure 3. Application Services

The business logic tier is the heart of the application where the application specific processing and business rules are maintained. Business logic placed in components bridge the client environments and the data tiers. The Windows DNA application platform has been developed through years of innovation around supporting high-volume, transactional, large-scale application deployments and provides a powerful runtime environment for hosting business logic components. As depicted in figure 3, the application platform for developing Windows DNA applications include web services, messaging services and component services.

Web Services

Integrated with Microsoft's application platform is a high-performance gateway to the presentation tier. Microsoft's Internet Information Server (http://www.microsoft.com/iis) enables the development of web-based business applications that can be extended over the Internet or deployed over corporate Intranets. With Internet Information Server, Microsoft introduced a new paradigm to the Internet - transactional applications. Transactions are the plumbing that makes it possible to run real business applications with rapid development, easy scalability and reliability.

Active Server Pages (ASP), a component of Internet Information Server, is the language neutral, compile-free, server-side scripting environment that is used to create and run dynamic, interactive Web server applications. By combining DHTML, scripting, and components, ASP enables application developers to create dynamic interactive Web content and powerful Web-based applications.

With the trend toward distributed computing in enterprise environments, it is important to have flexible and reliable communication among applications. Businesses often require independent applications that are running on different systems to communicate with each other and exchange messages even though the applications may not be running at the same time. Applications built using a combination of Active Server Page scripts communicating with cooperating components can interoperate with existing systems, applications and data.

Component Services

In the early 1990's, the underlying concept that facilitated interoperability was componentization; the underlying technology that enabled interoperability was Microsoft's Component Object Model (COM). As it turns out, componentization is not only a great way to achieve interoperability, but a great way to design and develop software in general. So in the mid-1990's Microsoft broadened COM's applicability beyond the desktop application to also include distributed applications by introducing Microsoft Transaction Server (MTS). MTS was an extension to the COM programming model that provided services for the development, deployment, and management of component-based distributed applications. MTS was a foundation of application platform services that facilitated the development of distributed applications for the Windows platform in a much simpler, more cost effective manner than other alternatives.

COM+ is the next evolutionary step of COM and MTS. The unification of the programming models inherent in COM and MTS services make it easier to develop distributed applications by eliminating the tedious nuances associated with developing, debugging, deploying, and maintaining an application that relies on COM for certain services and MTS for others. The benefits to the application developer is to make it faster, easier, and ultimately cheaper to develop distributed applications by reducing the amount of code required to leverage underlying system services.

To continue to broaden COM and the services offered today in MTS 2.0, COM+ consists of enhancements to existing services as well as new services to the application platform. They include:

Bring Your Own Transaction. COM components are able to participate in transactions managed by non-COM+ transaction processing (TP) environments that support the Transaction Internet Protocol (TIP).

Expanded Security. Support for both role-based security as well as process access permissions security. In the role-based security model, access to various parts of an application is granted or denied based on the logical group or *role* that the caller has been assigned to (e.g. administrator, full-time employee, part-time employee). COM+ expands on the current implementation of role-based security by including method-level security for both custom and IDispatch(Ex)-based interfaces.

Centralized Administration. The Component Services Explorer, a replacement for today's MTS Explorer and DCOMCNFG presents a unified administrative model, making it easier to deploy, manage, and monitor n-tiered applications by eliminating the overhead of using numerous individual administration tools.

In-Memory Database. For maintaining consistent durable state information and transient state information in a consistent manner. The In-Memory Database is an in-memory fully transactional database system designed to provide extremely fast access to data on the machine that it resides.

Queued Components. For asynchronous deferred execution when cooperating components are disconnected. This is in addition to the session based, synchronous client/server-programming model, where the client maintains a logical connection to the server today.

Event Notification. For times when a loosely coupled event notification mechanism is desirable. COM+ Events is a unicast/multicast, publish/subscribe event mechanism that allows multiple clients to "subscribe" to events that are "published" by various servers. This is in addition to the existing event notification framework delivered with connection points.

Load balancing. Allowing component-based applications to distribute their workload across an application cluster in a client-transparent manner.

Interoperating with Existing Systems, Applications and Data

As companies extend to embrace the Internet, offering new services and making information more readily available to it's customers, partners and employees, it is important that an application architecture provide the necessary means to extend inside the corporation to existing applications and data. Some software vendors have advocated that by wrapping your existing applications and data in a common component model in one language that runs all platforms you can achieve interoperability between applications and data. These are lofty goals that the industry has been promised over the past 30 years that fail to come to fruition. Microsoft's approach has been to reach out to existing applications and data on other platforms without disturbing the platform that the application or data resides on. Microsoft's application platform enables application developers to interoperate with applications and data through the use of Messaging Services, COM Transaction Integrator and through a framework called Universal Data Access.

Messaging Services

Microsoft Message Queue Server (MSMQ) (http://www.microsoft.com/msmq) provides loosely coupled and reliable network communications services based on a messaging queuing model. MSMQ makes it easy to integrate applications, by implementing a *push-style* business event delivery environment between applications, and build reliable applications that work over unreliable but cost-effective networks. The simple application based on the Component Object Model lets developers focus on business logic and not on sophisticated communications programming. MSMQ also offers seamless interoperability with other message queuing products, such as IBM's MQSeries, through products available from Microsoft's ISV partners.

Extending to the Mainframe Transaction Processing World

Using Microsoft's COM Transaction Integrator (TI), application developers can extend and expose CICS and IMS transaction programs through the use of COM components. COM Transaction Integrator consists of a set of development tools and runtime services that automatically "wrap" IBM mainframe transaction and business logic as COM components that run in a Windows DNA environment. All COM TI processing is done on a Windows NT Server; host communication is accomplished through the SNA Server (http://www.microsoft.com/sna). COM TI does not require the mainframe to run any executable code or be programmed in any special way.

Universal Data Access

Universal Data Access is Microsoft's strategy for providing access to information across the enterprise (http://www.microsoft.com/data). Today, companies building database solutions face a number of challenges as they seek to gain maximum business advantage from the data and information distributed throughout their corporations. Universal Data Access provides high-performance access to a variety of information sources, including relational and non-relational data, and an easy to use programming interface that is tool and language independent.

Universal Data Access does not require expensive and time-consuming movement of data into a single data store, nor does it require commitment to a single vendor's products. Universal Data Access is based on open industry specifications with broad industry support, and works with all major established database platforms.

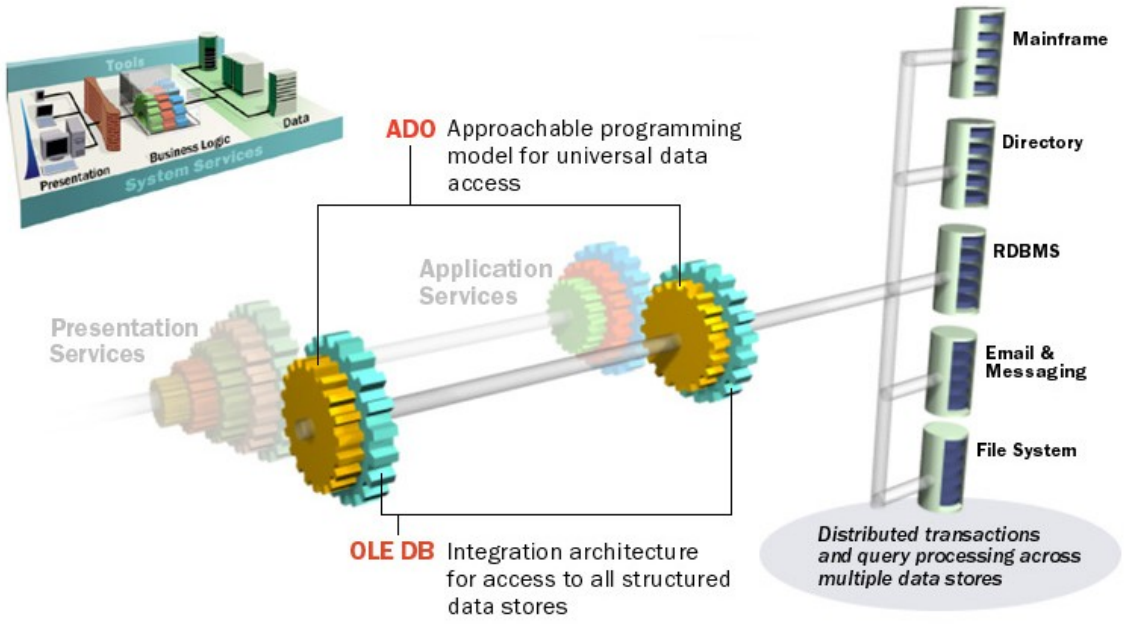

Figure 4. Data Access

As depicted in figure 4, the Universal Data Access based framework operates at two levels. At the systems level, OLE DB defines a component-based architecture specified as a set of COM based interfaces that encapsulate various database management system services. The OLE DB architecture does not constrain the nature of the data source; as a result, Microsoft and Independent Software Vendors have introduced providers for a wide variety of indexed sequential files, groupware products and desktop products (http://www.microsoft.com/data/oledb/products/product.htm). At the application level, ActiveX Data Objects (ADO) provides a high level interface to enable developers to access data from any programming language.

Conclusion

The Microsoft Windows DNA Architecture and the Windows NT platform offer many distinct advantages to customers and its ISV partners. Its key benefits are: Provides a comprehensive and integrated platform for distributed applications freeing developers from the burden of building the required infrastructure or assembling it using a piecemeal approach. Easily interoperates with existing enterprise applications and legacy systems to extend current investments. Makes it faster and easier to build distributed applications by providing a pervasive component model, extensive pre-built application services and a wide choice of programming language and tools support.