Microsoft Mi

crosoft Component Services

Server Operating System

A Technology Overview

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

04

CONTENTS

INTRODUCING MICROSOFT COMPONENT SERVICES

Every creator of enterprise software has the same goal: to produce the most effective applications in the least time. And every one of those creators knows that exploiting the services available in their environment is a key part of achieving that goal. As new innovations arise, application developers flock to them, creating new paradigms for software development.

There is no better example of this phenomenon than the tremendous popularity of Microsoft® component services. These services—the Component Object Model (COM) and Distributed COM, Microsoft Transaction Server, Microsoft Internet Information Server, and Microsoft Message Queue services in Windows NT® Server—focus on building better enterprise applications. By combining the best of today's standard technologies with new and innovative approaches, Microsoft has created a powerful, yet easy-to-use platform for building distributed applications. And as part of the Windows NT operating system, this application platform is available today to millions of developers. As a result, Microsoft's component services are becoming a foundation for a new generation of enterprise applications.

The Component Object Model (COM) and Distributed COM (DCOM)

Component-based development has become the software concept *du jour*, with its own conferences, magazines, and consultants. But the original component technology, and the one that is by far the most widely used, is Microsoft's Component Object Model (COM). Introduced in 1993, COM is now a mature foundation for component-based development, and it is the rare application for Windows® and Windows NT that doesn't use COM in some way. With its integrated services, and its excellent tool support from Microsoft and others, COM makes it easy to develop powerful component-based applications.

From its original application on a single machine, COM has expanded to allow access to components on other systems. Distributed COM (DCOM), introduced in 1996, makes it possible to create networked applications built from components. Available on various versions of UNIX, IBM mainframes, and other systems, DCOM is used today in applications ranging from cutting edge medical technology to traditional accounting and human resources systems. Once a bleeding edge technology, distributed components have gone mainstream, and the primary technologies enabling this are COM and DCOM.

Microsoft Transaction Server (MTS)

Building robust server applications that can scale to the enterprise is no mean feat. While some developers are capable of creating the necessary infrastructure services to accomplish this, it makes far more sense to provide these services in a standard way. This is exactly what the Microsoft Transaction Server does in Windows NT Server. By providing support for scalability, security, transactions, and more, MTS solves a range of fundamental problems, allowing enterprise developers to focus on the business problems at hand. And by offering its services through a remarkably easy-to-use programming model, MTS makes what might be a guru-only technology into a mainstream business tool.

Shipping since 1996, MTS was the first commercial software to combine transactions with components. In fact, every MTS-based application is built from COM components, and clients can access those applications remotely via DCOM. Other key MTS features include:

- **Automatic transactions** - MTS introduced the innovation of automatic transactions, which allow configuring a component's transactional requirements when that component is deployed. The result is a much greater potential for reuse of business objects built as MTS components.
- **Configurable security** - By allowing an administrator to define roles, then specify which interfaces and components can be accessed by clients in each role, MTS greatly simplifies the work required to create secure server applications.
- **Database connection pooling** - Components can reuse existing connections to a database rather than recreating new ones, greatly improving application performance and scalability.
- **Support for multiple databases and resource managers** - MTS-based applications can access SQL Server™, Oracle, and DB2 databases, as well as other resource managers such as Microsoft Message Queue.
- **Automatic thread support** - Application developers can write single-threaded components, then let MTS assign threads to those components as needed.
- **Component state management** - Components must give up any in-memory state when each transaction ends. This makes it easier to build correct applications while still allowing efficient resource sharing. MTS also provides the Shared Property Manager (SPM) for components that wish to store and later retrieve their in-memory state.
- **Process isolation through packages** - Individual applications can be grouped into one or more packages, and each package can run in its own process. This allows greater fault tolerance, since the failure of a single component will bring down only the package that component is part of.
- **Integration with mainframe transactions** - Through the COM Transaction Integrator, MTS transactions can initiate and control CICS transactions on IBM mainframes.
- **A broad range of development tools** - MTS allows developers to build appli-

cations in any of today's popular languages, including Microsoft® Visual Basic®, Java, C++, Cobol, and others.

MTS provides a rich set of integrated services, all focused on making it easy to build scalable, transaction-oriented, server-side components and applications. And best of all, all of these services are shipping today.

For additional information on MTS, you can access the MTS Reviewers Guide on the COM Web site at http://www.microsoft.com/com/mts/revguide-f.htm.

Microsoft Internet Information Server (IIS) and Active Server Pages (ASP)

Web technology has been the great software success story of the last ten years. Web-based applications have become an indispensable part of the distributed world, and so platforms for building them have become indispensable, too. Microsoft Windows NT Server's Internet Information Server (IIS), accessible from popular browsers, provides all the standard services one expects from a Web server: support for HTTP, Secure Sockets Layer (SSL), common gateway interface (CGI), and more. But because it is a part of Microsoft component services, IIS offers much more.

Along with support for standard CGI applications, for example, IIS also allows the creation of Active Server Pages (ASP). An ASP contains a simple program—a script—written in a simple language such as Microsoft® Visual Basic® Scripting Edition (VBScript) and executed at the server. ASPs offer developers many benefits, including:

- **Access to COM objects** - Because an ASP script can create and use COM objects, the entire world of component-based applications is accessible to it.
- **Integration with MTS-based applications** - Since MTS is built on COM, it's easy to access transactional applications from ASP scripts.
- **Transactional ASP scripts** - It's also possible to load MTS directly into IIS, creating ASP scripts that are themselves transactional—all operations performed by the script can succeed or fail as a unit.
- **Process isolation** - ASP scripts can run in the same process as IIS, offering the best performance, or in a separate process, offering greater isolation in the event of failures.
- **Support for standard, well-known languages** - Since VBScript is a subset of Visual Basic, millions of developers essentially know it today. ASP scripts can also be written in JavaScript, another popular and well-supported language.

Combining standard Web services with the innovations of components and transactions, as is done with ASP scripts, is an excellent example of the power provided by Microsoft component services.

Microsoft Message Queue (MSMQ)

DCOM provides synchronous communication using remote procedure calls, while Web-based applications rely on HTTP. But many distributed applications need the non-blocking, asynchronous communication provided by message queuing. Microsoft Message Queue (MSMQ) provides exactly this kind of service in Windows NT Server. With MSMQ, an application can send messages to another application without waiting for a response (in fact, the target application might not even be running). Those messages are sent into a queue, where they are stored until a receiving application removes them. If a response is expected, the sender can check a response queue at its leisure—there's no obligation to block waiting for a message. Message queuing is a flexible, reliable approach to communication, one that's appropriate for many kinds of applications.

MSMQ is very rich technology. Among its most important features are:

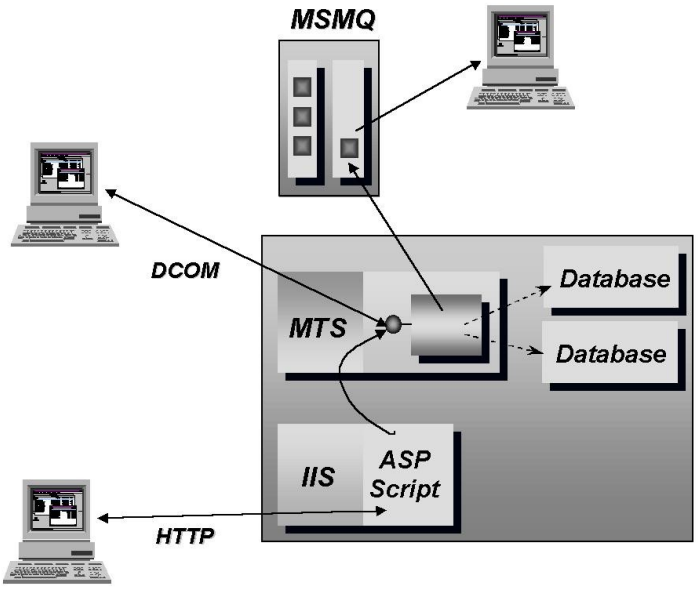
- **COM-based access** - The services of MSMQ can be accessed through a simple interface provided by COM components. This makes it straightforward to send and receive messages from within an ASP script, an MTS-based application, or any software that can use COM.
- **Integration with MTS** - Because MSMQ supports transactions, MTS-based applications can include the act of sending or receiving a message in a larger atomic unit. All of the operations in that transaction will succeed or fail as a group.
- **Automatic message journaling** - If requested, MSMQ will keep copies of messages sent or received by applications. Journals provide audit trails and can also make recovering from some kinds of failure easier.
- **Automatic notification** - If requested, MSMQ can notify a sending application that messages were (or were not) received and processed correctly. This service lets sending applications know when they can treat messages as delivered or, when failures occur, when they must take corrective action.
- **Built-in data integrity, data privacy, and digital signature services** - MSMQ can digitally sign and encrypt messages for transfer across the network. This protects messages from being viewed or changed during transmission (even when sent over public networks such as the Internet), and ensures that servers do not receive messages from unauthorized senders.
- **Message priority support** - MSMQ allows assigning priorities to messages and queues, then routes and delivers messages based on these priorities. Priorities let applications handle the most important messages first.
- **Support for multiple platforms** - MSMQ is available on many operating systems, including various versions of UNIX, IBM mainframes, and more, through Microsoft partner Level 8 Systems. Level 8 also provides gateways to existing messaging products such as IBM MQ Series.

By providing the flexible, reliable style of communication that message queuing makes possible, then linking this to other platforms, MSMQ plays a critical role in Microsoft component services.

For additional information on MSMQ, you can access the MSMQ Reviewers Guide at http://www.microsoft.com/NTServer/Basics/TechPapers/RevGuides.asp.

Putting the pieces together: An Example

The easiest way to understand how Microsoft component services work together is to look at an example. Suppose a catalog marketing firm chooses to build a Windows NT-based order entry system using Microsoft component services: COM/DCOM, MTS, IIS with Active Server Pages, and MSMQ. The diagram below shows how this might look.



The heart of the system is one or more COM components that contain the business logic necessary to fill orders. These order entry components access databases to determine whether there is sufficient inventory available to fill a particular order, and they must be able to simultaneously handle many order requests. Accordingly, these components are built to run under MTS, which provides multi-database transactions, scalability services, and more.

When an order has been completed, an order entry component sends a message to a warehouse indicating that the order should be packaged and shipped. Since there's no need for the component to wait for the order to be sent off, this request is

made using MSMQ. An application in the warehouse then extracts orders from a queue when convenient and ensures that they are filled.

But how are orders created? In this system, there are two different user interfaces to the order entry components. Like most catalog companies, this organization has operators who take phone calls from customers placing orders. These operators sit at workstations running Microsoft® Windows® 98 or Windows NT operating systems and use a custom client to create orders. This client communicates directly with the order entry components on the server using DCOM.

This organization also allows customers to place orders directly over the Internet. To allow this, an ASP script has been written that is accessed through IIS. A customer using any Web browser can visit the company's Web site, access this ASP script, and submit an order. Because ASP scripts can easily use COM objects, a customer ordering via the Web relies on the same MTS components for order entry, as does the DCOM client. All that changes is how that component is accessed.

As this example illustrates, all of the Microsoft component services work together to provide a complete solution, with each technology playing its role. The result is a textbook example of synergy: the whole is much greater than the sum of the parts.

Looking to the future: COM+

The evolution of Microsoft component services continues with COM+. By enhancing and extending existing services, COM+ further increases the value these services provide. COM+ includes:

- **A publish and subscribe service** - Provides a general event mechanism that allows multiple clients to “subscribe” to various “published” events. When the publisher fires an event, the COM+ Events system iterates through the subscription database and notifies all subscribers.
- **An in-memory database, with support for transactions** - The IMDB provides an application with fast access to data, without incurring the overhead associated with storing and accessing durable state to and from physical disk.
- **Queued components** - Allows clients to invoke methods on COM components using an asynchronous model. Such as model is particularly useful in on unreliable networks and in disconnected usage scenarios.
- **Dynamic Load balancing** - Automatically spreads client requests across multiple equivalent COM components.
- **Full integration of MTS into COM** - Includes broader support for attribute-based programming, improvements in existing services such as Transactions, Security and Administration, as well as improved interoperability with other transaction environments through support for the Transaction Internet Protocol (TIP).

COM+ builds on what already exists—it is not a revolutionary departure. Microsoft component services provide an infrastructure for building enterprise applications, and enterprises seldom welcome revolutions in their infrastructure. But software technology cannot stand still. The goal, then, must be to provide useful innovations that make it easier to create great applications without disrupting what's already in place. By extending and further unifying the existing component services, COM+ does exactly this.

Taken as a whole, Microsoft component services provide a powerful, flexible, and easy-to-use platform for building distributed applications. Nothing else available offers the same level of integration, broad tool support, and solid services.