微软编程圣典丛书（影印版）

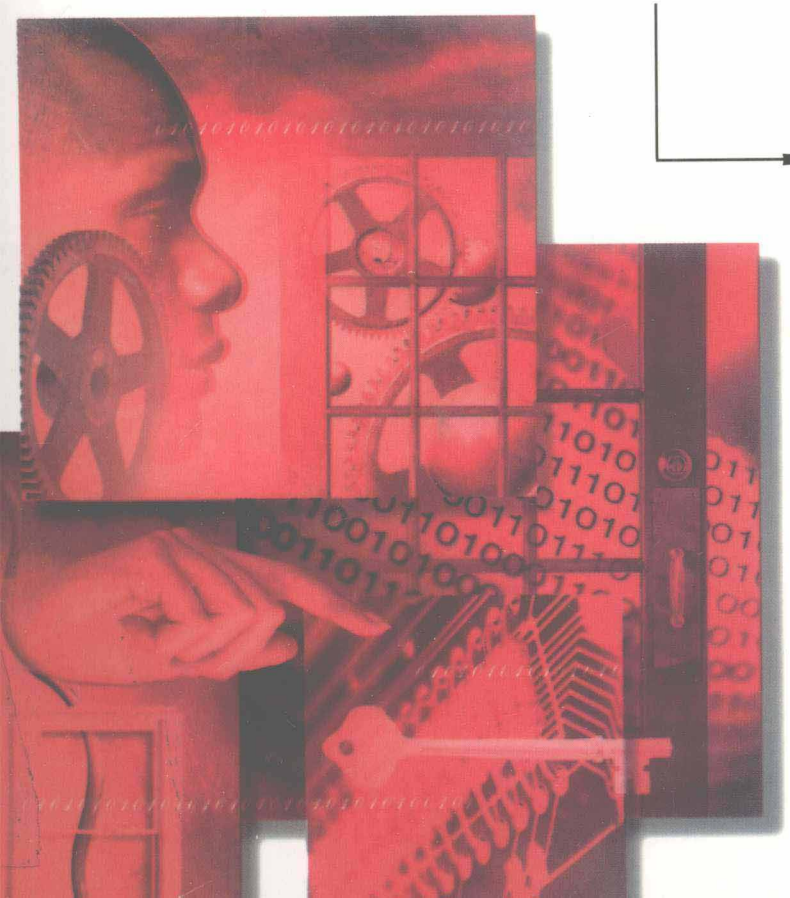
Microsoft

配光盘

COM+ 基本服务内幕

（影印版）

Inside COM+ Base Services



- 来自微软的第一手技术资料，深入专业的编程技术
- 揭示 COM+的强大功能，开发企业商务解决方案
- 高级程序员必备

[美] Guy Eddon
Henry Eddon 著

北京大学出版社
http://cbs.pku.edu.cn

微软编程圣典丛书(影印版)

COM+ 基本服务内幕

Microsoft 公司 著

北京大学出版社

内 容 简 介

本书是《微软编程圣典丛书（影印版）》之一，讲述如何用 Windows COM+（组件对象模型）开发基于组件的商务解决方案，内容涉及 Windows DNA 体系结构、接口定义语言、线程等。为了增加本书的实用性，特通过配套光盘提供了丰富的程序实例以及本书的电子版。

本书由微软公司组织专家编写，具有相当的技术深度，是中、高级程序员必备的参考书。

书　　名：COM+ 基本服务内幕（影印版）

责任著作者：Microsoft 公司 著

标 准 书 号：ISBN 7-900629-37-8/TP・31

出　版　者：北京大学出版社

地　　址：北京市海淀区中关村北京大学校内　100871

网　　址：http://cbs.pku.edu.cn

电　　话：出版部 62752015　发行部 62765127　62754140　编辑室 62765127

电 子 邮 箱：wdzh@mail.263.net.cn

印　刷　者：北京大学印刷厂印刷

发　行　者：北京大学出版社

经　销　者：新华书店

787 毫米×1092 毫米　16 开本　42.375 印张　1200 千字

2000 年 9 月第 1 版　2000 年 9 月第 1 次印刷

定　　价：108.00 元

丛 书 序

世纪交替，IT 产业更加步履匆匆。

Microsoft 公司早已以其在编程方面的非凡成就闻名于世，并树立了在计算机软件领域和发展史上不可动摇的地位。毋庸置疑，该公司技术上的优势是其获得成功的重要因素之一。今天，它的技术不但已经变得非常强大，而且具有惊人的发展速度。尤其是 Windows 2000 技术的推出，更是展示了 Microsoft 的无穷魅力，它突然间提供了如此丰富的新特性，使我们仿佛在一瞬间便被淹没在 Windows 2000 浩瀚的技术海洋之中！

工欲善其事，必先利其器。作为 Windows 应用程序设计人员，必须紧密跟踪 Microsoft 公司的最新技术，深入 Microsoft Windows 编程的内幕，掌握关键的编程技术。这套《微软编程圣典丛书（影印版）》的推出，就是为了向有关的专业人员全面推介微软编程的核心技术，以便于他们设计高质量的 Windows 应用程序。

Microsoft 技术博大而精深，绝非某个人在短时间内所能掌握。为此，特按照技术上的逻辑关系组织成 9 个相对独立的部分，分别涉及基于服务器的应用程序、COM+基本服务、Windows 网络编程、国际化程序、MFC、Windows 编程、服务器端应用程序、Outlook 与 Exchange 编程、驱动程序模型等。每一部分的内容独立成册，集中讲述一组相关的编程技术。这套《微软编程圣典丛书（影印版）》共 9 本。特定编程领域的专业人员可以从中选取自己需要的一本或几本，使学习过程更加快速、省时、有效而直观。

这套丛书中的任何一本都涉及一门完整的编程技术，因此有着相当的深度，而且内容比较丰富。为了避免将其写成深奥而抽象的理论书，特在书中适当的位置穿插进许多贴切的程序实例。另外，每本书都有配套的 CD-ROM，内有书中的程序实例和本书的电子版。

本套丛书由 Microsoft 公司组织相关领域的专家编写。他们深谙 Microsoft 的编程技术内幕，具有丰富的程序开发经验，所以，这套丛书是他们智慧的结晶，是该领域极具权威性的著作，堪称独领风骚。

鉴于此，特向中、高级 Windows 应用程序设计人员郑重推荐这套佳作！

出版者

2000 年 9 月

FOREWORD

Well, it's been over a decade since I started writing applications for Microsoft Windows. Back then, it was easy for someone to say that they really understood Windows and all of its subsystems. Each of these subsystems exposed only a few hundred functions: in Windows 2.11, Kernel had 283, User had 141, and GDI had 213, for a grand total of 637 functions. How hard could it be to understand what these few functions did?

Over the past 10 years, Microsoft has greatly extended these modules and has added numerous subsystems to Windows: telephony, remote access, print spoolers, 3-D graphics, Internet access, security, registry, services, multimedia, networking, and so on. It is now impossible for any individual to fully understand the entire operating system. For this reason, I've advised developers to pick certain components of the system, study them, learn them, and become specialists in them. This is not to say that you should ignore other parts of the system—I've specialized in Kernel and User, but I've also dabbled in GDI, networking, and lots of other areas.

At Microsoft, different teams develop each of these subsystems and each team develops its own "philosophy." For example, I know that the registry functions all start with a *Reg* prefix and return an error code. But I also know that most Kernel functions have no special prefix and return *FALSE* if they fail; in these cases, I have to call *GetLastError* to see the reason for the failure. This inconsistent behavior among subsystems is one of the reasons that Windows programming has had a reputation for being difficult to learn.

By now, it is obvious to everyone that Microsoft is firmly committed to Windows for the present and for the foreseeable future. To achieve this end, it has needed a plan that allows new subsystems to be added without steepening the developer's learning curve too much. In other words, there must be consistency in using the various subsystems (or components). The technology to address this need is COM+.

Microsoft exposes new technologies by implementing each new subsystem as a COM+ object. You can easily see this design when you use directory services (Active Directory), transaction services (Microsoft Distributed Transaction Coordinator), graphics (DirectX), shell extensions, controls (ActiveX controls), data (OLE DB), scripting (ActiveX scripting), and on and on. COM+ is now *the* way to interact with these subsystems. In fact, in an effort to reduce

the Windows learning curve for new developers, Microsoft is going back to older subsystems and exposing those systems as COM+ objects.

It is now imperative that Windows developers understand the core infrastructure of COM+. With this understanding, you can easily take advantage of these new subsystems as well as more easily expose your own subsystems. Distributed computing is a massive undertaking that requires addressing many difficult problems, such as data transfer, incompatible computer architectures, disparate network architectures, timing issues, and so on. Fortunately, Microsoft has teams of developers working to solve these problems and to make things easier for the rest of us. Windows 2000 offers the first enterprise-level release of COM+.

With Microsoft firmly behind it, COM+ will undoubtedly be the best way to interact with subsystems, not only on a single machine but also on any computer anywhere in the world (and possibly beyond). *Inside COM+ Base Services* is our guidebook. Let's keep our fingers crossed and hope that it is translated into Martian.

Jeffrey Richter

PREFACE

COM+ is not a radical departure from COM—it is the next stage of evolution of the COM programming model. COM was originally designed as a minimalist's component architecture. With the advent of the three-tier programming model, applications have become more complex. To help developers who work in this new world, COM+ offers a richer set of services than was available in COM. These services evolved from the technology previously known as Microsoft Transaction Server (MTS) and include features such as automatic transactions, role-based security, load balancing, object pooling, queued components, the in-memory database, and an external publish-and-subscribe event model.

COM+ offers many powerful and useful run-time services that save you the effort of building similar services yourself. While working on this book, we came to realize that it is impossible to explain these services without first explaining the fundamental component model at the heart of COM+. Whether or not you use these services in your components—for some developers, COM+ services offer little advantage[1]—you must understand the fundamental COM+ programming architecture before you can use COM+ effectively.

Inside COM+ Base Services describes the fundamental component model at the core of COM+. Once you understand the issues involved in building software components, you'll be able to decide where and how to use the COM+ component services effectively. *Inside COM+ Component Services* (forthcoming from Microsoft Press) covers these services in detail, along with issues involved in building multitier enterprise applications.

In this book, we make two assumptions—that you're interested in learning about COM+ and that you're familiar with a modern programming language such as C++, Microsoft Visual Basic, or Java. One of the central tenets of COM+ is the concept of language neutrality, so we've structured *Inside COM+ Base Services* around that idea. Although you can build COM+ components in a wide variety of development environments, we focus on the most popular trio: C++, Visual Basic, and Java. Visual Basic and Java developers will learn a lot from this book even though these languages hide a large part of the COM+ infrastructure.

---

1. For example, DirectX, a set of specialized components for graphics programming in Microsoft Windows, does not use any COM+ component services because of its critical performance requirements.

In many cases, we present examples in C++ and then show how components written in higher-level languages can tie into the same functionality. However, you must understand the fundamentals of C++ in order to use many of the sample programs.

Remember the Sojourner rover that roamed the surface of Mars during the summer of 1997? It ran on an 8-bit Intel 80C85 processor containing only 6500 transistors (compared with 5.5 million transistors in a Pentium). It had a radio modem capable of 9600 bps and was powered by solar energy and nonrechargeable lithium D-cell batteries. It was a feat of modern engineering achieved with the most basic components. In a less dramatic way, COM+ is like that—it consists of a fundamental set of ideas that can give rise to some amazingly powerful systems.

System Requirements for the Companion CD

To run the code on the companion CD, you will need Microsoft Visual Studio 6.0 and Microsoft Windows 98 or Microsoft Windows NT 4.0. (Microsoft Windows 2000 is required for some chapters.)

Acknowledgments

Thanks to Eric Stroo, our acquisitions editor, for guiding yet another book to a successful conclusion. Mary Kirtland and Saji Abraham of Microsoft Corporation supported us throughout this project by answering many questions. Thanks to Eric Maffei and the rest of the MSJ gang. Thanks to Alice Turner, the project editor; Marc Young, the technical editor; and Ina Chang, the manuscript editor, for their hard work and dedication to this book.

Guy Eddon
Henry Eddon
http://www.guyeddon.com

CONTENTS