



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** 著
Henry Eddon

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

江苏工业学院图书馆
藏书章

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。

Copyright (2000) by Microsoft Corporation

Original English language edition Copyright © 2000 (year of first publication by author)

By Microsoft Corporation (author)

All rights published by arrangement with the original publisher, Microsoft Press, a division of Microsoft Corporation, Redmond, Washington, U.S.A.

著作权合同登记号：图字 01-2000-3029 号

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 advantage¹—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

<i>Foreword</i>	xiii
<i>Preface</i>	xv

PART I: FUNDAMENTAL PROGRAMMING ARCHITECTURE

CHAPTER ONE

Component Software	3
From Object-Oriented Programming to Component Software	6
Object-Oriented Programming	7
Code Sharing and Reuse	8
Component Software	9
The Evolution of COM+	11
From OLE to COM+	12
RPC and COM+	15
From COM to COM+	16
Windows DNA: A Three-Tier Approach	18
Component Services	20

CHAPTER TWO

The <i>IUnknown</i> Interface	29
Interface Definition Language	31
The Client	36
The <i>CoInitializeEx</i> Function	36
The <i>CoCreateInstance</i> Function	36
The Methods of <i>IUnknown</i>	39
The <i>CoUninitialize</i> Function	42
The V-Table Situation	43
Building the Client Project	46
The Component	48
Implementing the <i>AddRef</i> and <i>Release</i> Methods	49
Implementing the <i>IUnknown::QueryInterface</i> Method	50

The <i>ISum::Sum</i> Method (Finally)	57
The <i>IClassFactory</i> Interface	57
Exported DLL Functions	63
The <i>CoCreateInstance</i> Function Revisited	67
Building the Component Project	70
Component Registration	74
Merging Object Identity	81
Containment	83
Aggregation	85
CHAPTER THREE	
Language Integration	91
Type Libraries	92
Using Type Libraries	92
Building a Type Library	93
Registering a Type Library	96
An Easy C++ Client	101
C++ Templates (A Quick Introduction)	102
Namespaces	110
The Active Template Library	111
The ATL COM AppWizard	112
The ATL Object Wizard	113
Adding Methods and Properties to an Interface Using ATL	115
Building a Simple COM+ Object Using ATL	116
COM+ Programming in Visual Basic	118
<i>QueryInterface</i> : The Visual Basic Way	119
Building a Client in Visual Basic	122
Implementing COM+ Interfaces in Visual Basic	122
Building a Component in Visual Basic	124
COM+ Programming in Java	126
Calling a COM+ Object from Java	132
Implementing COM+ Objects in Java	136
ActiveX Controls and JavaBeans Integration	141
The Sandbox Model	142
CHAPTER FOUR	
Apartments	145
A Quick Review of Threads	146

Apartment Types	147
Single-Threaded Apartments	149
Multithreaded Apartments	157
Marshaling Interface Pointers Between Apartments	158
How to Choose a Threading Model	162
Threading Models for In-Process Components	165
Apartment Interactions	166
Objects That Support the MTA Model	169
Objects That Support All Apartment Models	170
The Free-Threaded Marshaler	172
Neutral Apartments	181
Comparing the Apartment Models	183
Writing Thread-Safe Components	185
Apartment and Language Integration	187
Threading Options for Visual Basic Components	188
Threading Options for Java Components	192
The Ten Threading Commandments	193

PART II: BASE FACILITIES

CHAPTER FIVE

Automation	197
The <i>IDispatch</i> Interface	199
Automation Types	200
Implementing <i>IDispatch</i>	212
Designing a Pure Dispinterface	212
Designing a Dual Interface	213
Implementation Techniques	217
Properties	225
Collections	227
The (New and Improved) <i>IDispatchEx</i> Interface	229
Building Automation Clients	235
Building Automation Clients in C++	235
Building Automation Clients in Visual Basic	240
Scripting	242
Building Automation Clients in Script	243
Scriptlets	245

CHAPTER SIXExceptions **249**

Error Codes	250
<i>FACILITY_ITF</i> Error Codes	251
Helper Macros	251
Rich Error Information	252
The <i>ISupportErrorInfo</i> Interface	253
The <i>ICreateErrorInfo</i> Interface	253
Obtaining Error Information	255
The <i>IErrorInfo</i> Interface	256

CHAPTER SEVENComponent Categories **259**

Standard Component Categories	261
Default Components	262
Registering a Component Category	265
The <i>ICatRegister</i> Interface	265
Obtaining Component Category Information	268
The <i>ICatInformation</i> Interface	268

CHAPTER EIGHTConnection Points **271**

A Simple Connectable Object	272
The Source Interface	272
The <i>ICorrelationPoint</i> Interface	273
The <i>ICorrelationPointContainer</i> Interface	276
Implementing a Sink in C++	279
A Visual Basic Sink	283
A Java Sink	289
A Complete Connectable Object	291
Enumerators	292
When to Use Connection Points	297

CHAPTER NINEType Information **301**

Creating a Type Library	302
Adding Type Information	305
Obtaining Type Information	318
The <i>ITypeLib</i> Interface	319

The <i>ITypeInfo</i> Interface	321
The <i>ITypeComp</i> Interface	325
Reading Type Information Using High-Level Languages	326

CHAPTER TEN

Persistence **329**

The <i>IPersist</i> Interface Family	329
The <i>IStream</i> Interface	331
Persisting an Object	333
Implementing a Persistable Object	335
Building Persistable Objects in Visual Basic	339
Building Persistable Objects in Java	344
Structured Storage	346
The <i>IStorage</i> and <i>IStream</i> Interfaces	347
The <i>IPropertySetStorage</i> and <i>IPropertyStorage</i> Interfaces	349

CHAPTER ELEVEN

Monikers **355**

Initializing Objects	355
Class Objects	357
Custom Activation Interfaces	357
More on Monikers	358
The <i>IMoniker</i> Interface	359
The <i>MkParseDisplayName</i> Function	364
The Class Moniker	367
The Marvelous Moniker: Improving the Class Moniker	370
The New Moniker	377
The Java Moniker	378
The Running Object Table	379

PART III: REMOTING ARCHITECTURE

CHAPTER TWELVE

Surrogates **385**

DLL Surrogates	386
Running In-Process Components Locally	386
Running Components Remotely	388
Custom Surrogates	391
A Custom DLL Surrogate: <i>DIINanny</i>	391

An Introduction to Marshaling	394
Standard Marshaling	396
Type Library Marshaling	398
Custom Marshaling	399
 CHAPTER THIRTEEN	
Executable Components 401	
Building an Executable Component	403
Registering the Class Objects	407
Remote Instantiation	409
Integrating the Marshaling Code	411
Managing the Lifetime of an Executable Component	414
Race Conditions	415
Executable Component Shutdown	417
Custom Activation Interfaces	420
Singletons	424
 CHAPTER FOURTEEN	
Custom Marshaling 427	
Marshaling Interface Pointers: An Overview	428
Re-Creating an Interface's V-Table	429
Interprocess Communication	430
Will That Be Custom or Standard Marshaling?	431
Can You Say "Custom Marshaling"?	435
Pardon Me, What Is the CLSID of Your Proxy Object?	437
How Big Did You Say Your Interface Is?	438
Unmarshaling the Interface Pointer	444
Marshal-by-Value	450
 CHAPTER FIFTEEN	
Standard Marshaling 455	
The Standard Marshaling Architecture	456
The Standard Marshaling Interfaces	460
Registering the Proxy/Stub DLL	475
Converting Marshaled Interface Pointers to Strings	476
The OBJREF Moniker	478
Handler Marshaling	479

CHAPTER SIXTEEN

Interface Definition Language	485
Types	485
Enumerated Types	487
Directional Attributes	487
Arrays	491
Fixed Arrays	492
Conformant Arrays	492
Varying Arrays	495
Open Arrays	496
Character Arrays	498
Multidimensional Arrays	501
Passing Arrays of User-Defined Types from Visual Basic	502
Pointers	503
Full Pointers	504
Unique Pointers	504
Reference Pointers	505
Interface Pointers	505
Interface Design Recommendations	509

CHAPTER SEVENTEEN

Asynchronous Calls	511
Making Asynchronous Calls	511
Defining Asynchronous Interfaces	512
Calling Asynchronous Interfaces	513
Implementing Asynchronous Interfaces	516
Interoperability	518
Call Cancellation	519
Requesting Method Call Cancellation	520
Terminating the Method	522
Pipes	523

CHAPTER EIGHTEEN

Security	527
The Windows Distributed Security Model	527
The COM+ Security Model	528
COM+ Security Packages	531

Declarative Security: The Registry	532
Default Security	533
Configuring Default Access and Launch Permissions	537
Configuring Component Security: The AppID Key	539
The <i>IAccessControl</i> Interface	541
Configuring Component Identity	547
Programmatic Security	550
The <i>ColInitializeSecurity</i> Function	551
Using the <i>IAccessControl</i> Interface with <i>ColInitializeSecurity</i>	560
Activation Credentials: The COAUTHINFO Structure	563
The <i>IServerSecurity</i> Interface	566
Cloaking	570
The <i>IClientSecurity</i> Interface	572
 CHAPTER NINETEEN	
The Network Protocol 575	
Spying on the Network Protocol	577
Running Network Monitor	580
Remote Activation	581
Internet Services	583
Calling All Remote Objects	586
The ORPCTHIS and ORPCTHAT Structures	588
Marshaled Interface Pointers	590
The Standard Object Reference	592
The DUALSTRINGARRAY Structure	593
The <i>IRemUnknown</i> Interface	595
The <i>IRemUnknown2</i> Interface	600
The OXID Resolver	600
Garbage Collection	604
A Remote Method Call	607
Channel Hooks	608
A Useful Channel Hook: Obtaining the Client's Name	611
 APPENDIX	
Remote Procedure Calls 617	
The Design and Purpose of RPC	617
Interface Definition Language	618
Binding	619

Location Transparency	619
Handles	620
The Prime Application	621
Client Initialization	622
Client Computation	625
The Prime Server	625
Context Rundown	626
Debugging	627
Distributed Computation	627
<i>Bibliography</i>	629
<i>Index</i>	631