Preface

This book discusses COM+ component services. Each service is covered in its own chapter, and each chapter discusses a similar range of issues: the problem the service addresses, possible solutions to that problem, an in-depth description of the COM+ solution, tradeoffs, design, and implementation guidelines, tips, and known pitfalls. I have tried to provide useful design information and lessons I learned while applying COM+. I also describe COM+ helper classes and utilities I developed that will enhance your productivity significantly. (The COM+ Events helper objects and the COM+ Logbook are prime examples.) This book focuses on the "how to"—that is, it provides practical information. You should read the chapters in order, since most chapters rely on information discussed in the preceding chapters. The book also aims to explain COM+ step by step. A software engineer already familiar with COM who wants to know what COM+ is and how to use it can read this book and start developing COM+ applications immediately.

Scope of This Book

Here is a brief summary of the chapters and appendixes in this book:

- Chapter 1 introduces the Component Services Explorer and basic COM+ terminology. This chapter deliberately holds your hand as you develop your first "Hello World" COM+ component. Subsequent chapters do much less handholding and assume you are familiar with the COM+ environment. If you already have experience with basic COM+ development, feel free to skip this chapter.
- Chapter 2 demystifies the COM+ context by presenting it as the key mechanism for providing component services using call interception. Generally, you need not be concerned with contexts at all. However, the COM+ context underlies the way COM+ services are implemented.
- Chapter 3 describes two scalability-enabling mechanisms that COM+ provides for a modern enterprise application: object pooling and Just-in-Time Activation (JITA). The discussion of instance management, and especially JITA, is independent of transactions. Early COM+ documentation and books tended to couple instance management and transactions. However, I found that not only can you use instance management independently of transactions, but it is easier to explain it that

way. Besides explaining how to best use object pooling and JITA, Chapter 3 describes other activation and instance management COM+ services such as the constructor string.

- Chapter 4 explains the difficult, yet common, problems that transactions address, and provides you with a distilled overview of transaction processing and the transaction programming model. The difficult part of writing this chapter was finding a way to convey the right amount of transaction processing theory. I want to help you understand and accept the resulting programming model, but not bury you in the details of theory and COM+ plumbing. This chapter focuses on COM+ transaction architecture and the resulting design considerations you have to be aware of.
- Chapter 5 first explains the need in the component world for a concurrency model and the limitations of the classic COM solution. It then describes how the COM+ solution, activities, improves deficiencies of apartments.
- Chapter 6 shows how to access component and application configuration information programmatically using the COM+ Catalog interfaces and objects. Programmatic access is required when using some advanced COM+ services and to automate setup and development tasks. This chapter provides you with comprehensive catalog structure diagrams, plenty of sample code, and a handy utility.
- Chapter 7 explains how to secure a modern application using the rich and powerful (yet easy to use) security infrastructure provided by COM+. This chapter defines basic security concepts and shows you how to design security into your application from the ground up. You can design this security by using COM+ declarative security via the Component Services Explorer and by using advanced programmatic security.
- Chapter 8 explains what COM+ queued components are and how to use them to develop asynchronous, potentially disconnected applications and components. In addition to showing you how to configure queued components, this chapter addresses required changes to the programming model. If you have ever had to develop an asynchronous method invocation option for your components, you will love COM+ queued components.
- Chapter 9 covers COM+ loosely coupled events, why there is a need for such a service, and how the service ties into other COM+ services described in earlier chapters (such as transactions, security, and queued components). Many people consider COM+ events their favorite service. If you have had to confront COM connection points, you will appreciate COM+ Events.

- Chapter 10 shows how .NET components can take advantage of the component services described in the previous chapters. If you are not familiar with .NET, I suggest you read Appendix C first—it contains an introduction to .NET and C#. Chapter 10 repeats in C# many of the C++ or VB 6.0 code samples found in earlier chapters, showing you how to implement them in .NET.
- Appendix A helps you develop a useful and important utility—a flight recorder that logs method calls, errors, and events in your application. Logging is an essential part of every application and is especially important in an enterprise environment. The logbook is also an excellent example of the synergies arrived at by combining multiple COM+ services. It is also a good representation of the design approaches you may consider when combining services.
- Appendix B describes the changes, improvements, and enhancements introduced to COM+ in the next release of Windows, Windows XP. Instead of writing the book as if Windows XP were available now (as of this writing it is only in beta), I chose to write the book for the developer who has to deliver applications today, using Windows 2000. When you start using Windows XP, all you need to do is read Appendix B—it contains the additional information you need.
- Appendix C describes the essential elements of the .NET framework, such as the runtime, assemblies, and how to develop .NET components. The appendix allows a reader who is not familiar with .NET to follow Chapter 10.

Some Assumptions About the Reader

I assume that you are an experienced COM developer who feels comfortable with COM basics such as interfaces, CoClasses, and apartments. This book is about COM+ component services, not the component technology used to develop a COM/DCOM or .NET component. You can still read the book without this experience, but you will benefit more by having COM under your belt. I assume you develop your components mostly in C++ and ATL and that you write occasional, simple client code in Visual Basic. I also use trivial C# in Chapter 10 to demonstrate how .NET takes advantage of COM+ services, but you don't need to know C# to read that chapter. A .NET developer should also find this book useful: read and understand the services in Chapter 1 through Chapter 9, and then use Chapter 10 as a reference guide for the syntax of .NET attributes.

Definitions and Text Conventions

The following definitions and conventions apply throughout this book:

- A component is an implementation of a set of interfaces. A component is what you mark in your IDL file (or type library) with CoClass or a class in C#.
- An object is an instance of a component. You can create objects by calling `CoCreateInstance( )` in C++, specifying the class ID (the type) of the object you want to create. If you use Visual Basic 6.0, you can create objects using `new` or `CreateObject( )`. A C# client uses `new` to create a new instance of a component.
- I use the following terms in the book: CoCreating refers to calling `CoCreateInstance()` in C++, or `new` or `CreateObject( )` in Visual Basic. Querying an object for an interface refers to calling `IUnknown::QueryInterface( )` on the object. Releasing an object refers to calling `IUnknown::Release( )` on the object.
- The graphical notations in Figure P-1 are used in almost every design diagram in the book. The "lollipop" denotes an interface, and a method call on an interface is represented by an arrow beginning with a full circle.

Figure P-1. Interface and method call graphical notations

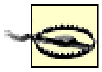
- Error handling in the code samples is rudimentary. The code samples serve to demonstrate a design or a technical point, and cluttering them with too much error handing would miss the point. In a production environment, you should verify the returned HRESULT of every COM call, catch and handle exceptions in C#, and assert every assumption.

I use the following font conventions in this book:

- *Italic* is used for new terms, citations, online links, filenames, directories, and pathnames.
- `Constant width` is used to indicate command-line computer output and code examples, as well as classes, constants, functions, interfaces, methods, variables, and flow-controlled statements.

- **`Constant-width bold`** is used for code emphasis and user input.
- *`Constant-width italic`* is used to indicate replaceable elements in code statements.

This icon indicates a note or tip.



This icon indicates a warning.

Other COM+ Books and References

This book describes how to use COM+ component services in your application. It focuses on how to apply the technology, how to avoid specific pitfalls, and design guidelines. If you want to know more about COM+ in general and the nature of component technology, I recommend the following two books that helped me a great deal in my attempt to grasp COM+.

COM+ and the Battle for the Middle Tier by Roger Sessions (John Wiley & Sons, 2000) is hands down the best "why" COM+ book. It explains in detail, with excellent examples and in plain language, the need for software components and component services. For example, instead of the page or two this book includes on the motivation for using transactions, Sessions devotes two fascinating chapters to the topic. The book goes on to compare existing component technologies (such as COM, CORBA, and Java) and their corresponding suites of component services. It also contains a few case studies from real-life systems that use COM+. Roger Sessions also has a unique way of eloquently naming things—providing the most appropriate term, which is often not the name Microsoft uses. Whenever it makes sense, this book uses Sessions' terminology, such as "instance management" instead of the Microsoft term "activation."

Understanding COM+ by David S. Platt (Microsoft Press, 1999) is probably the best "what" COM+ book. The book describes the services available by COM+ and provides sidebar summaries for the busy reader. It is one of the first COM+ books, and Platt worked on it closely with the COM+ team.

I also used the MSDN Library extensively, especially the "Component Services" section, while writing this book. Although the information in this library tends to be terse, the overall structure is good. Use this book to learn how to apply COM+ productively and

effectively, and use the MSDN Library as a reference for technical details and a source for additional information.

How to Contact Us

We have tested and verified the information in this book to the best of our ability, but you may find that features have changed (or even that we have made mistakes!). Please address comments and questions concerning this book to the publisher:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international/local)
(707) 829-0104 (fax)

The web site for the book lists examples, errata, and plans for future editions. You can access this page at:
http://www.oreilly.com/catalog/comdotnetsvs
To ask technical questions or comment on this book, send email to:
bookquestions@oreilly.com
Or to me directly:
juval.lowy@componentware.net
For more information about our books, conferences, software, resource centers, and the O'Reilly Network, see our web site:
http://www.oreilly.com

Acknowledgments

A book is by no means the product of just the author's work. It is the result of many events and individuals, like links in a chain. I cannot possibly name everyone, ranging from my parents to my friends. I am especially grateful for my two friends and colleagues, Marcus Pelletier and Chris W. Rea. Marcus worked with me on large COM+ projects, and together we confronted the unknown. Marcus's thoroughness and technical expertise is a model for every programmer. Chris's comments and insight into a reader's mind have contributed greatly to this book's accuracy, integrity, and flow. I wish to thank Yasser Shohoud for verifying my approach to transaction processing and sharing with me his own, Richard Grimes for reviewing the book, and Roger Sessions for writing the Foreword. Thanks also to Johnny Blumenstock for providing me with a place to write. Finally, this book would not be possible without my