Pro SQL Server 2005 Service Broker

Klaus Aschenbrenner

Apress®

Pro SQL Server 2005 Service Broker

Copyright © 2007 by Klaus Aschenbrenner

ISBN-13 (pbk): 978-1-59059-842-9

ISBN-10 (pbk): 1-59059-842-3

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editors: James Huddleston, Dominic Shakeshaft
Technical Reviewer: Fabio Claudio Ferracchiati
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Jeff Pepper, Dominic Shakeshaft, Matt Wade
Project Manager: Kylie Johnston
Copy Edit Manager: Nicole Flores
Copy Editor: Nicole Abramowitz
Assistant Production Director: Kari Brooks-Copony
Production Editor: Laura Esterman
Compositor: Pat Christenson
Proofreader: Lori Bring
Indexer: Carol Burbo
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Source Code/Download section.

For Karin.
Every day that starts with you turns out to be a great day.
I will always love you.

Contents at a Glance

Foreword xiii
About the Author xv
About the Technical Reviewer xvii
Acknowledgments xix
Introduction xxi

PART 1 ■■■ The Service Broker Programming Model

■**CHAPTER 1** Fundamentals of Message-Based Processing 3
■**CHAPTER 2** Introduction to Service Broker. 17
■**CHAPTER 3** Service Broker in Action. 31
■**CHAPTER 4** Service Broker Activation. 69
■**CHAPTER 5** Service Broker with Managed Code 119
■**CHAPTER 6** Locking and Transaction Management 151
■**CHAPTER 7** Distributed Service Broker Applications. 221

PART 2 ■■■ Advanced Service Broker Programming

■**CHAPTER 8** Advanced Distributed Service Broker Programming 253
■**CHAPTER 9** Service-Oriented Database Architecture 285
■**CHAPTER 10** Real-World Application Scenarios. 329
■**CHAPTER 11** High Availability and Scalability 395
■**CHAPTER 12** Administration 423

■**INDEX** 437

Contents

Foreword . . . xiii
About the Author . . . xv
About the Technical Reviewer . . . xvii
Acknowledgments . . . xix
Introduction . . . xxi

PART 1 ■■■ The Service Broker Programming Model

■CHAPTER 1 Fundamentals of Message-Based Processing . . . 3

Message Concepts . . . 4
- Message Anatomy . . . 4
- Messaging in Daily Life . . . 5

Why Messaging? . . . 5
- Asynchronous Message Processing . . . 6
- Deferred Message Processing . . . 6
- Fault Tolerance . . . 7
- Distributed Systems . . . 7

Messaging Problems . . . 8
- Performance . . . 8
- Queue Reader Management . . . 9
- Transaction Management . . . 10
- Message Sequencing and Correlation . . . 10
- Maintenance . . . 11

Messaging Architectures . . . 12
- SOA . . . 12
- SODA . . . 12

Available Messaging Technologies . . . 13
- MSMQ . . . 13
- Queued Components . . . 14
- BizTalk Server . . . 14
- XML Web Services . . . 14
- WCF . . . 14

Summary . . . 15

CHAPTER 2 Introduction to Service Broker 17

Conversations 17
- Dialogs 17
- Dialog Lifetime 18
- Conversation Groups 19
- Message Sequencing 20
- Reliable Delivery 20
- Error Handling 21

Anatomy of a Service 21
- Message Types 23
- Contracts 23
- Queues 24
- Service Programs 25
- Routes 25

Security 25
- Transport Security 25
- Dialog Security 26

Message Processing 27
Performance 29
Benefits 30
Summary 30

CHAPTER 3 Service Broker in Action 31

Defining Service Broker Applications 31
- Message Types 32
- Contracts 36
- Queue 39
- Service 41

Sending Messages 45
Retrieving and Processing Messages 51
- Retrieving Messages 53
- Processing Messages 55

Error Handling 59
- Error Handling in Service Programs 59
- Poison-Message Handling 63
- Ending Conversations with Errors 66

Summary 68

CHAPTER 4 Service Broker Activation 69

Activation Basics 69
- Startup Strategies 70
- When Is Activation Needed? 71

Internal Activation 72
Controlling Message Throughput. 79
Stored-Procedure Signing 82
Calling a Stored Procedure in Another Database 90
Using a Single Stored Procedure to Process Many Queues 94
External Activation. 97
Parallel Activation 112
Troubleshooting Activation 117
When the Activated Stored Procedure Doesn’t Run 117
When Messages Remain on the Queue. 117
Summary 118

CHAPTER 5 **Service Broker with Managed Code** 119

The Managed Assembly 119
Architecture and Design of the Managed Assembly. 123
Building a Managed Service Broker Client 129
Building a Managed Service Program. 133
Derive Your Service Class 134
Implement an Entry Point. 134
Implement Message Type Handling 136
Deploy the Assembly 136
Register the Managed Stored Procedure 138
Configure the Service Broker Activation 139
Using the Service Program 139
A Practical Example. 142
Summary 150

CHAPTER 6 **Locking and Transaction Management** 151

Conversation Groups and Locks 151
State Handling 158
GET CONVERSATION GROUP 159
The Receive Loop with State Handling 160
State Handling with a Managed Stored Procedure 164
A Practical Example 170
Compensation Logic with Service Broker 198
Transaction Management 204
Basic Receive Loop 204
Measuring Performance. 205
Batched Commits. 208
Cursor-Based Processing 210
Set-Based Processing 213
Binary Payload 216
Summary 219

CHAPTER 7 Distributed Service Broker Applications 221

Communication 221
- Service Broker Protocols 222
- Sending a Message 222

Routing. 224
- Routing Algorithm 225
- Managing Routes 226

Distributed Applications. 228
- The Application 228
- Setting Up Routes 229
- Establishing a Communication Channel 232
- Setting Up Security 236

Summary 249

PART 2 Advanced Service Broker Programming

CHAPTER 8 Advanced Distributed Service Broker Programming 253

Transport Security 253
- LOCAL Route 254
- TRANSPORT Route 255

Dialog Security 257
- Service Broker Security Protocol 257
- Configuration 259

Encryption 268
- Transport Encryption 269
- Dialog Encryption 270
- Recommendation 271

Transport Protocol 271
- Setting Up Tracing 271
- The Captured Service Broker Message 273
- Replaying Service Broker Messages 276

Replacing Certificates 276
- Transport Security 276
- Dialog Security 277

Service Listing Manager 277
- Exporting a Service Listing 279
- Importing a Service Listing 281

Summary 284

CHAPTER 9 Service-Oriented Database Architecture 285

Service-Oriented Database Architecture. 285
SOA. 286
Reasons for SODA Architectures 287
Requirements for a SODA Service Provider. 288
Data in SODA. 288
Outside Data. 289
Inside Data 290
SODA Features in SQL Server 2005 292
XML Support. 292
Native Web Services 295
SQLCLR 307
Query Notifications. 316
Summary 327

CHAPTER 10 Real-World Application Scenarios 329

Asynchronous Triggers 329
Defining the Problem 330
Implementing the Trigger 331
Creating the Service Broker Infrastructure 333
Writing the Service Program 334
Batch Frameworks 337
Creating the Service Broker Infrastructure 338
The Implementation of the Batch Framework. 339
Extending the Batch Framework 345
Publish-Subscribe Frameworks 348
Defining the Infrastructure. 349
Applying Publisher Logic 351
Publishing Information. 356
Priority-Based Message Processing 358
Implementing Priority-Based Messaging. 358
Reliable Web Service Requests. 371
Service Broker Infrastructure. 372
Implementation of the Web Proxy 376
Using the Web Proxy in a Smart Client 388
Summary 393

CHAPTER 11 **High Availability and Scalability** 395

Database Mirroring 395
  Implementation Details 396
  Setting Up Database Mirroring. 397
  Using Service Broker with Database Mirroring 402
Load Balancing 404
  Service Deployment 405
  Initiator Configuration. 406
Message Forwarding 408
  Reliable Delivery. 408
  Security 408
  Network Topology Abstraction. 409
  Centralized Routing Instance 410
  Work Distribution 410
  Using Message Forwarding 411
  Monitoring Message Forwarding 415
Configuration Notice Service 416
  Implementing Dynamic Routing. 417
  Implementing the Configuration Notice Service 417
Summary 422

CHAPTER 12 **Administration** 423

SQL Profiler 423
  Using SQL Profiler 425
System Monitor 428
SQL Server Management Objects 432
  Creating Service Broker Objects 434
  Retrieving Information 435
Summary 436

INDEX 437

Foreword

Prior to my tenure at Microsoft, I spent eight years building distributed applications for a large telecommunications firm. These applications were distributed across a wide range of platforms and written in a variety of languages, spanning Visual Basic to COBOL. There was no common object model or even a common programming language that was shared across these applications and the machines they ran upon. The applications were maintained by different administrators in unrelated departments with disparate business requirements. In a few rare cases, applications that shared data were not even running at the same time.

Faced with this reality, it quickly became clear that the raging debate between software vendors over which object-remoting strategy was best (COM+ vs. CORBA) was, to say the least, irrelevant. Without fail, these vendor offerings assumed a world wherein applications were built using an object-oriented language (Java or C++) on either UNIX or Windows, and that both the client and server were always "up" and ready to communicate.

Ironically, this dearth of vendor offerings empowered us to step back and question the status quo. Was object remoting really the best way to build distributed applications? Did the software vendors truly understand the nuances of building large-scale applications that spanned both trust and geographic boundaries?

To find the answers to these questions, we studied the nature of existing large-scale systems both within and outside of the world of computing: the postal system, the airline industry, and the banking industry. Without fail, we found systems that were very loosely coupled. Specifically,

- Communication was always (and only) done by passing data in agreed-upon formats (e.g., letters, checks).
- Communicating systems *never* stopped working as a result of one of their peers slowing down or stopping (i.e., a person can buy groceries even when the bank is closed).
- Throughput was not a function of the distance between the systems (i.e., you can send the same number of letters from Seattle per hour whether they are addressed to people also living in Seattle or in a location as remote as Tokyo).

The same was true of our business applications. Almost everywhere we looked we found applications that needed a framework to enable sharing data in asynchronous, autonomous ways vs. the synchronous, dependent model enabled by object remoting.

As a result of this research, we abandoned all object-remoting aspirations and began working with state-of-the-art queuing technologies such as MQSeries and the like in earnest, as they seemed the perfect fit for building loosely coupled applications. Unfortunately, our joy was short-lived as we discovered that state-of-the-art queuing forced developers to choose between reliable messaging that was slow and not-so-reliable messaging that was fast. In addition, the vast majority of our applications used a database in some fashion, and the close combination of queues with databases quickly left us wishing that our queues came with all the amenities that we had come to love in our database tables. Why couldn't we query our queues? Why did the queuing system run under different transactions? Why couldn't we back up the queues and relational tables together? For that matter, why didn't database systems include all of this queuing stuff anyway?

Five years later it was 1999 and I found myself employed by Microsoft and sitting in Adam Bosworth's office with the rare opportunity to do something about all of this. I had just completed a 45-minute sales pitch to Adam (who was then one of the managers of SQL Server), asking him to grant me three people for three months to prove that SQL Server and message queuing should become one. Adam, having never laid eyes on me until that meeting, looked at me and asked, “Why should I fund this? Tomorrow I leave for a one-month vacation, and until 45 minutes ago I’d never heard of you.” I answered, “Because you’d be a d**n fool not to. That’s why. If I’m wrong, fire me. If I’m right, we can change distributed applications forever. What are three people for three months to you anyway?” Apparently this reply resonated with Adam, as he not only funded the effort, but also helped sell it to the rest of the team.

Six years and many all-nighters later, we shipped SQL Service Broker in SQL Server 2005. Our original goals were simple:

- Make it fast. “If you can’t deliver 1,000 messages/second, exactly once, and in order, then don’t even ship it.”
- Integrate deeply with SQL Server. Queues should be backed up, failed over, and queryable just like relational tables.
- Simplify the programming model. Message queuing is too hard. Make it simpler.

In short, we met these goals and more. Whether your needs are just kicking off something to run in the background or processing more than 10,000 messages per second on a 16-CPU machine, SQL Service Broker may add value to your applications. (Yes, it really is that fast. Eat your heart out, MQSeries . . .).

SQL Service Broker is already being used in a variety of applications. Some people use it in localized ways—for example, to send messages for stored procedures to process later so that their web application can respond more quickly to browser-based clients. Others use it to speed up writing audit/log messages from within their applications. Still others use it to load data warehouses. Finally, there are those who build modern, Service-Oriented Applications (SOA) that span thousands of branch offices and their back-end data centers.

No vision of this size comes to pass without many individuals playing their part. Phase 1 was the work the Service Broker team did to build the product. Phase 2 consisted of the early adopters who had the insight not only to see the potential in the technology, but to help others see it as well. The book you now hold is the result of a lot of hard work from such an individual. You are about to become the lucky recipient of Klaus’s insight, hard work, and interaction with the SQL Service Broker development team. Klaus is not only a visionary, but also an all-around nice guy. It has been our pleasure to have him sit with us at lunch and co-present with us as at various conferences. He has become one of us, a partner in the vision that is Service Broker.

Now all that is left is for you, the reader, to take the next step and actually build applications.

Good luck!

Gerald Hinson (`geraldh@microsoft.com`)
Founder/Development Manager of SQL Service Broker

About the Author

KLAUS ASCHENBRENNER works as a software architect for ANECON (a Microsoft Gold Certified Partner for Data Management and ISV/Software Solutions) in Vienna, Austria. Klaus has worked with the .NET Framework and especially with SQL Server 2005 from the very beginning. In 2004 and 2005, Klaus was honored with Microsoft Most Valuable Professional (MVP) awards for his tremendous support of the .NET community. He currently travels around the world teaching clients the core concepts of SQL Server 2005. He also works very closely with Microsoft to drive the next version of SQL Server, code-named Katmai.

About the Technical Reviewer

■**FABIO CLAUDIO FERRACCHIATI** is a senior consultant and a senior analyst/developer using Microsoft technologies. He works for Brain Force (`http://www.brainforce.com`) in its Italian branch (`http://www.brainforce.it`). He is a Microsoft Certified Solution Developer for .NET, a Microsoft Certified Application Developer for .NET, and a Microsoft Certified Professional, as well as a prolific author and technical reviewer. Over the past ten years, he has written articles for Italian and international magazines and coauthored more than ten books on a variety of computer topics.

Acknowledgments

It's 8:30 p.m., and I'm sitting here in a train between Vienna and Linz writing this introduction. Interestingly, my experience and success with SQL Server 2005 and especially with Service Broker also started in a train traveling between Vienna and Linz in the year 2002. That year, I bought a very interesting book about the next version of SQL Server code-named Yukon. It was called *A First Look at SQL Server 2005 for Developers*, and it was written by three well-known SQL Server experts: Bob Beauchemin, Niels Berglund, and Dan Sullivan. It was amazing for me to read about the completely new concepts in Yukon, such as the integration of the .NET Framework (SQLCLR) and the XML data type. The book also contained a chapter entitled "Service Broker," which I didn't really address at the time. Several months later I read this chapter completely, and then understood the basic concepts behind Service Broker.

When I realized what scenarios one could use Service Broker for, I started thinking faster and faster. I realized that Service Broker and all the other new technologies in Yukon were an important milestone in the history of SQL Server. In the years from 2002 to 2006, I worked a lot with SQL Server 2005. I was also one of those crazy people who tried out every new Community Technology Preview (CTP) of Yukon, and I moved my first SQL Server 2005 project into production based on a beta version of SQL Server 2005.

I also consulted for several clients around Europe on SQL Server 2005 and made many new contacts in the SQL Server community through the talks I gave at various developer conferences around the world. But in August 2006, my life put me in another direction. I got an email from an acquisition editor at Apress, Jim Huddleston, who asked me if I would write a book about Service Broker.

And yes, I took this chance and worked out the original table of contents during the whole next weekend (sorry about this, Karin). After I returned my proposal back to Jim, he was so crazy about the content that he just said I should start writing immediately. The only thing I needed was a technical reviewer, so I thought back to the guys who wrote *A First Look at SQL Server 2005 for Developers* and asked Bob Beauchemin if he would become my technical reviewer. He accepted within a few hours, and my writing started immediately. This was the early beginnings in August 2006 of my book about Service Broker.

Since August 2006, a lot of time has passed, and some good and bad things have occurred. I want to mention and thank the people around the whole world who helped me with their passion to put together this book on Service Broker. First of all, I want to thank my acquisition editor, Jim Huddleston, for his support and input on this great project. With very great sadness, I found out that Jim passed away on a Sunday in February 2007 from a heart attack. This was totally unexpected, because Jim was a healthy man. Unfortunately, I can't say this to you personally, Jim, but thanks for all your support and your power that you incorporated into this book, and I enjoyed meeting with you in Seattle.

As I've stated, Bob Beauchemin acted as my technical reviewer in the beginning, but unfortunately, Bob had to leave this role due to some time constraints. But Bob, I want to thank you for your great support, your comments, and your improvements on the first few chapters. I had a great time working with you. After Bob left as a technical reviewer, Apress was able to provide me with another technical reviewer within a few days: Fabio Claudio Ferracchiati. Fabio, thanks for your detailed and great technical reviews of each chapter. I know I've made your life very hard throughout these days, but we finally produced the greatest book about Service Broker. Thanks for your help on this!

During the writing, I also got great support from Microsoft in dealing with all my nasty and detailed questions about Service Broker. I especially want to mention Remus Rusanu, who answered each of my questions within a few hours. Remus also provided me with some samples for Chapter 6

and Chapter 11. Remus, thanks for your support and also for the in-depth flip-chart session during the 2006 PASS Community Summit in Seattle—I enjoyed your drawings and the direct view of Bill Gates' office. Furthermore, I also want to mention Gerald Hinson (who wrote the foreword), Rushi Desai, and Rick Negrin from the Service Broker team, who also provided me with additional information about Service Broker and who helped introduce me to some other smart people at Microsoft.

When I talk about the Service Broker team at Microsoft, there is also one person I must mention here: Roger Wolter. Roger is the lead architect on Service Broker who architected and designed all the things you'll learn throughout this book. Furthermore, Roger has also written the first book in the world about Service Broker: *The Rational Guide to SQL Server 2005 Service Broker*. Roger, thanks for this great introductory book on Service Broker—it helped me a lot in the beginning.

A great book can't be made without a publisher. The team at Apress helped me a lot to put everything together to write the best available book on Service Broker. First of all, there is my project manager, Kylie Johnston. Kylie, I hope we'll soon meet face to face, but it was just the greatest time working with you. Thanks for your help, for your directions on the book, and for your easy-going project plan. A big thank you also goes to Nicole Abramowitz for her help on the copy edits, and to Laura Esterman who acted as my production editor. Thanks for all your help, and I've enjoyed the time with both of you.

A big thank you also goes to my colleagues at ANECON for their support and their tremendous help during the last nine months. I want to thank them for their understanding those mornings when I came in either with just a few hours of sleep or directly from my writings and was often a little bit stressed.

Last but not least, I have to thank my family. A big thank you goes to my parents, Herbert and Dagmar. For the last 13 years, you've been the driving factor behind my passion of computers, computers, and computers. You supported me in every direction to move my career forward to the point where I'm standing now. Thanks for everything!

Finally, there's Karin, my girlfriend and the most important person in my life. Karin, I'm very, very amazed at how easy it was for you during the last nine months while I've been writing the book. There were so many evenings, weekends, and even weeks when I had no time, because Kylie (with her time-consuming project plan) drove my life. But you handled those days so easily, which gave me the power to concentrate on the book. Thanks for your love, your support, your passion, and your easy understanding of the endless nights. Thanks for everything and especially for your love. I'll dedicate this book to you. I will always love you.

Introduction

SQL Server 2005 Service Broker is an asynchronous messaging framework directly built into SQL Server 2005. In this book I show how you can use the power of Service Broker to program asynchronous, message-based, distributed, secure, and scalable database applications.

Who This Book Is For

This book is for database developers and application developers who want to learn about Service Broker and programming message-based applications with SQL Server 2005.

How This Book Is Structured

The book is divided into two parts:

Part 1: The Service Broker Programming Model: The first part of this book introduces you to the general concepts and programming APIs of Service Broker. After reading through this part, you'll be able to implement asynchronous, distributed, reliable, and secure Service Broker applications.

Part 2: Advanced Service Broker Programming: The second part of the book shows you the more advanced Service Broker features, including the internals of Service Broker and how to scale out Service Broker applications to any required size. I also talk to you about Service-Oriented Database Architecture (SODA), where Service Broker acts as one of the key pillars.

The first part of the book is composed of the following chapters:

Chapter 1: Fundamentals of Message-Based Processing: This chapter introduces you to the core concepts of message-based programming, as well as to some of the fundamental issues you'll encounter in this programming approach. Once you understand the theory, you'll examine how to work through issues with Service Broker in the next chapter.

Chapter 2: Introduction to Service Broker: This chapter introduces Service Broker from an architectural point of view and explains how Service Broker solves the problems discussed in Chapter 1.

Chapter 3: Service Broker in Action: This chapter teaches you how to program your first message-based application with Service Broker.

Chapter 4: Service Broker Activation: Now that you know the fundamentals of Service Broker, this chapter introduces you to the activation feature of Service Broker, which allows you to process incoming Service Broker messages automatically.

Chapter 5: Service Broker with Managed Code: This chapter shows how you can use the advantages of the SQLCLR to implement Service Broker applications directly with managed code.

Chapter 6: Locking and Transaction Management: As soon as you want to implement asynchronous, scalable, message-based applications, you must take care of locking strategies. This chapter introduces the Service Broker locking functionalities and also shows you how to write highly efficient Service Broker applications through different transaction-management strategies.

Chapter 7: Distributed Service Broker Applications: This chapter, which closes the first part of this book, teaches you how to distribute Service Broker applications to physically different machines.

The second part of the book is composed of the following chapters:

Chapter 8: Advanced Distributed Service Broker Programming: This chapter goes into the more technical details of distributed Service Broker applications and shows which options are available for your applications.

Chapter 9: Service-Oriented Database Architecture: Service-Oriented Database Architecture (SODA) is a new concept propagated by Microsoft where the database server—in this case, SQL Server 2005—acts as a full-blown application server. SODA consists of several pillars, and, as you'll see, Service Broker is one of those key pillars.

Chapter 10: Real-World Application Scenarios: This chapter details different real-world application scenarios where Service Broker can offer huge benefits and lead to better scalability.

Chapter 11: High Availability and Scalability: SQL Server 2005 is all about high availability and scalability. One of the best things about Service Broker is that you can use SQL Server's high-availability and scalability features directly for your Service Broker applications without any effort. This chapter shows you how to use these features.

Chapter 12: Administration: This final chapter teaches you how you can administer your Service Broker applications and which features are provided by Service Broker in this area.

Prerequisites

You will need SQL Server 2005 Standard Edition/Developer Edition and Visual Studio 2005 Standard Edition.

Downloading the Code

The source code for this book is available to readers at `http://www.apress.com` in the Source Code/Download section of this book's home page. Please feel free to visit the Apress website and download all the code there. You can also check for errata and find related titles from Apress.

Contacting the Author

You can reach Klaus at his website, `http://www.csharp.at`, and via his weblog, `http://www.csharp.at/blog`. You can send further questions to `Klaus.Aschenbrenner@csharp.at`.