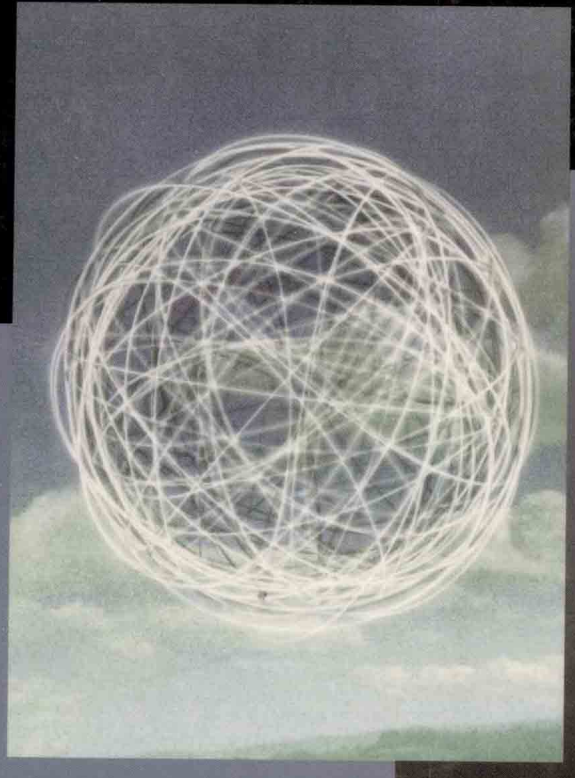
Using XML with Legacy Business Applications



Michael C. Rawlins

Using XML with Legacy Business Applications

Michael C. Rawlins

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

The publisher offers discounts on this book when ordered in quantity for bulk purchases and special sales. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact:

International Sales
(317) 581-3793
international@pearsontechgroup.com

Visit Addison-Wesley on the Web: www.awprofessional.com

Library of Congress Cataloging-in-Publication Data

Rawlins, Michael C.
Using XML with legacy business applications / Michael C. Rawlins.
p. cm.
ISBN 0-321-15494-0 (Paperback : alk. paper)
1. XML (Document markup language) 2. Business—Computer programs. I. Title.

QA76.76.H94R375 2003
005.7'2—dc21

2003010094

ISBN 0-321-15494-0
Text printed on recycled paper
1 2 3 4 5 6 7 8 9 10—CRS—0706050403
First printing, August 2003

Preface

Someday most business applications will be able to read and write XML files. Until that happens, you are probably going to need techniques and utilities like those presented in this book. This book is for people who use business applications that don't currently support XML and for people who develop those applications and want to build XML support into them. It is designed to provide a tool kit of techniques and utilities that can help you perform common enterprise application integration (*EAI*), business-to-business (*B2B*), or electronic data interchange (*EDI*) data conversion operations using XML.

Nothing in this book is rocket science. Any good programmer with experience in the relevant technologies could develop any of these techniques and utilities. The point of this book is that I've done it so that you don't have to. As is often said, good programmers develop good programs. Better programmers steal what they can and modify it. Steal this code.

If you are a user of a business application and already have or can afford to procure a capable EAI or EDI software package, you probably don't need this book. However, if you have only some simple needs that don't justify the purchase of such a package, or if for some other reason you don't want to or can't afford to spend thousands of dollars to purchase one, then this book is for you. And if you are interested in an open, nonproprietary, standards-based, and portable approach to data conversion, then this book is for you, too.

Acknowledgments

Many people helped in the production of this book. First, I want to thank my editors, Mary O'Brien and Shelley Kronzek. Without Shelley's enthusiasm and encouragement, I don't think this project ever would have gotten off the ground. Without Mary's help and persistence, I might not have finished yet. To Chrysta Meadowbrooke, thank you *very* much for your excellent copyediting. I also want to thank the rest of the staff at Addison-Wesley, particularly Brenda Mulligan for her help in many small ways.

My thanks to my technical reviewers—Eve Maler, Daniel K. Appelquist, Steve Vinoski, and Cameron Laird. Your candid comments helped make this a much better book. Innumerable thanks also to the Computer Science faculty at the University of Texas at Dallas, in particular to Dr. Lawrence Chung. Yes, you can teach an old dog new tricks. And thanks to many professional colleagues too numerous to mention by name, but especially those from the XML Forum of the Postsecondary Electronic Standards Council, ANSI ASC X12, the OASIS UBL Technical Committee, and the ebXML Working Group.

Finally, I want to thank my good wife, Diana, for her patience, support, and tolerance for *my* skewed priorities as I finished up this project.

Contents

Chapter 1

Introduction

You know what a "legacy application" is? It's one that works.—Bill Cafiero

This chapter introduces the approach presented in the book and discusses the business and technical rationales for it. Based on these, the overall features of the approach and the major design decisions are reviewed. The chapter concludes with a description of the primary audiences and suggestions for how to use the book.

The Problem

So, what's the problem? You probably picked up this book (or are browsing this sample chapter on the Web) because the title sounded like it might have something to do with your situation. The Preface probably fleshed out that impression a bit more. By now you probably think you have the book pretty well scoped out, and you may not be that far from the truth. But to save you a bit of time, let me be very specific about the problem (or, more accurately, the problems) that this book addresses.

The primary problem is this: You have one business application that imports and exports data in XML formats, and one that doesn't. You need to make these two applications talk to each other. The XML-enabled application may be yours, or it may belong to someone with whom you (or your application's users) need to exchange data. (Imagine an important customer or government agency sending

you a letter that says, "You will receive orders from us in XML format by January 15 or be assessed a $50 penalty for each paper order.") The legacy application is most probably yours. However, the shoe may be on the other foot: Your application speaks XML, but the other guy's application doesn't. He expects *you* to deal with it. (Imagine an important customer sending you a letter that says, "You will receive orders from us by EDI by January 15 or be assessed a $50 penalty for each paper order.")

There are really two perspectives to this primary problem, and this book addresses both. According to one perspective, the application developer should just fix the stupid application so it can speak XML. From an end user's viewpoint, this is perfectly reasonable, and I'll deal with the problem from this perspective.

From the other perspective, whoever developed the application can't or won't support XML by the time you need it (which might have been last week). There may be several reasons for this. The vendor of a commercial package may not see sufficient market demand. The vendor may have retired the product or, even worse, gone out of business. If the application was developed in-house, the original developers may be gone, retired, or dead. So, there are quite a few reasons why you might need to come up with a solution on your own. In this book I'll also deal with the problem from this perspective.

That, in about 400 words, sums up the primary problem this book addresses. However, in solving this problem from the perspective of someone who needs to come up with a solution on their own (as opposed to that of the developer who just needs to fix the stupid application), we find that the solution might apply to similar problems. Why is that? Well, if we have a method to go from a legacy flat file to XML and a method to go from XML to a legacy flat file, we have a method to go from a legacy flat file to a different legacy flat file (with XML in between). The same thing holds for other common formats such as comma-separated values (*CSV*) or Electronic Data Interchange (*EDI*). By coming up with a solution to the primary problem, we find that we have a general solution for all sorts of file format conversion problems. That is also what this book is about.

Before we leave the problem definition and start addressing the solution, there are just a couple more points we need to clarify. First, what, exactly, is a legacy application? Aside from Mr. Cafiero's pithy observation, for the purposes of this book we'll refer to a legacy application as any working application that doesn't currently provide native support for XML (that is, the application can't produce XML documents as output and consume XML documents as input).

Second, why use XML with a legacy application? That is a reasonable question, but answering it is beyond the scope of this book. In this book I assume that you have already answered that question for yourself. You're going to use XML and you have figured out why; all you want to know is *how*. There has been enough good general material written on the benefits of application integration and electronic commerce that I feel there's very little I can add for justification.

In regard to XML in particular, I assume that you have one application that uses XML and a legacy application that doesn't, and that you need to integrate them. To do that you need either to convert between XML and the format required by the legacy application or to build native XML support into the legacy application. You know what you need to do and why you need to do it. This book is intended to help you with how to do it.

What Do We Need in a Solution?

When we ask, "What do we need?" we're talking about requirements. There are two types of requirements: functional and nonfunctional (the latter are also known as quality requirements or system constraints). The former have to do with what the system is supposed to do. The latter have to do with how it does what it does. Both are important, and both determine the overall approach of this book.

Beyond the overall dictate of solving the problem, two distinct sets of requirements are imposed on the solution by technical end users on the one hand and by application developers on the other. I'll talk a little later about why I'm dealing with both, but for now if you don't care about the other group you can just skip the relevant paragraphs.

Functional Requirements

The technical end user who has an application that doesn't speak XML more than likely needs the solution to do one or more of the following:

- Convert an XML-formatted file to a flat file
- Convert a flat file to an XML-formatted file
- Convert an XML-formatted file to a CSV file
- Convert a CSV file to an XML-formatted file
- Convert an EDI-formatted file to an XML-formatted file
- Convert an XML-formatted file to an EDI-formatted file

A user may want the solution to support other formats, but CSV, flat file, and EDI should handle most cases. For example, an end user may also need to get data out of a database (relational, hierarchical, or otherwise) and put it into an XML format, or go back the other way. Sorry, but these types of problems are a bit beyond our scope. I will, however, give in Chapter 12 an overview of some approaches for doing things like this. When I present the approaches, you'll understand why problems like this exceed our scope a bit.