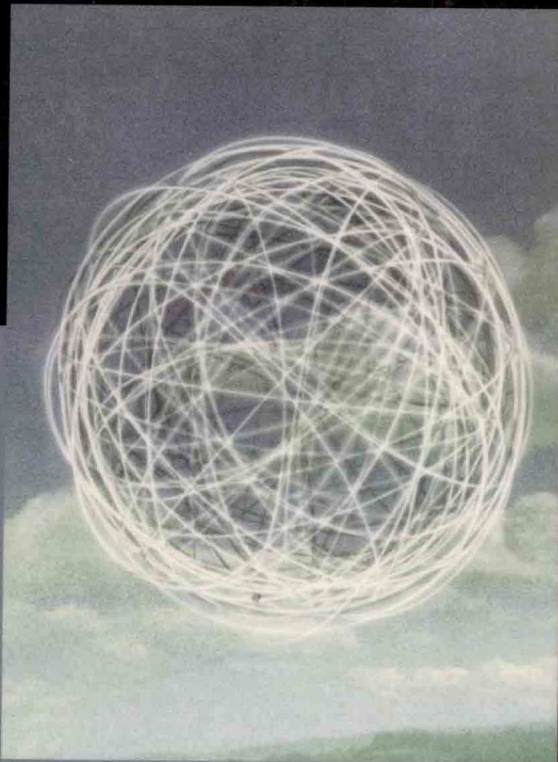


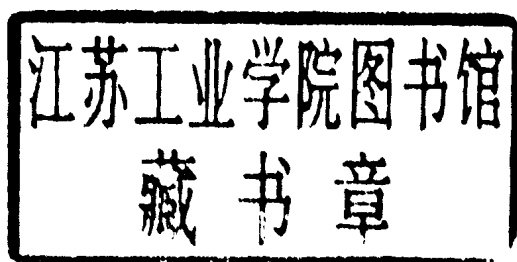
Using XML with Legacy Business Applications



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

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Unless otherwise noted, the opinions presented in this book are those of the author. They should not be construed as the positions of any organization with which he may be affiliated.

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

Copyright © 2004 by Pearson Education, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher. Printed in the United States of America. Published simultaneously in Canada.

For information on obtaining permission for use of material from this work, please submit a written request to:

Pearson Education, Inc.
Rights and Contracts Department
75 Arlington Street, Suite 300
Boston, MA 02116
Fax: (617) 848-7047

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

Preface	xvii
Chapter 1 Introduction	1
The Problem	1
What Do We Need in a Solution?	3
Functional Requirements	3
Nonfunctional Requirements: Good, Fast, and Cheap	4
The Overview of a Solution	6
Architecture	7
Why Not Use XSLT for Everything?	10
Two Implementations of the Architecture: Java and C++	13
The Document Object Model	14
Why Use the DOM?	15
How to Use This Book	16
Chapter Organization	17
Notes for Primary Audiences	18
Chapter Summaries	18
Conventions	21
What You Need to Use This Book	22
What You Should Already Know	22
Web Site and Contact Information	22
General Software	23
Java Software	24
C++ Software	25

For Developers	26
General Coding Approach and Conventions	26
Additional C++ Considerations	28
How You Can Use the Utilities and Code	29
References	31
Resources	32
 Chapter 2	 33
Converting XML to CSV	
Requirements	33
Running the Utility	34
Sample Input and Output	36
Design	38
Naming Elements	38
Module Logic	40
Java Implementation	45
main in XMLToCSVBasic.java	46
write in CSVRowWriter.java	47
Error Handling	49
C++ Implementation	52
main in XMLToCSVBasic.cpp	53
write in CSVRowWriter.cpp	55
Error Handling	57
Comparing the Java and C++ Implementations	59
Enhancements and Alternatives	60
Validation of the Input Document	61
Using a ColumnNumber Attribute	61
A Recursive Algorithm	61
Advanced Functionality	63
Resources	64
 Chapter 3	 65
Converting CSV to XML	
Requirements	65
Running the Utility	66
Sample Input and Output	67
Design	70
Main Routine	70
CSVRowReader Class	72
Java Implementation	75
main in CSVToXMLBasic.java	75
parse in CSVRowReader.java	78
write in CSVRowReader.java	78
C++ Implementation	78
main in CSVToXMLBasic.cpp	79

parse in CSVRowReader.cpp	81
write in CSVRowReader.cpp	81
Comparing the Java and C++ Implementations	81
Enhancements and Alternatives	82
Validation of the Output Document	82
Advanced Functionality	82
Some Observations	82
Resources	85

Chapter 4	Learning to Read XML Schemas	87
	Rope, Anyone?	87
	A Few Different Document Approaches	89
	DTD Refresher	91
	Foundation Concepts and Terminology	93
	Elements and Types	93
	Simple and Complex	94
	General Features	95
	Schema Declaration in Instance Documents	98
	Comments and Documentation	98
	Element Declarations	99
	Simple Content Elements	99
	Schema Built-in Data Types	100
	Extending and Restricting Simple Types	102
	Setting a Maximum Length	102
	Setting Minimum and Maximum Values	103
	Patterns for Identifiers	103
	Code Lists	104
	Attribute Declarations	105
	Complex Content Elements	107
	Types of Content	108
	Mandatory, Optional, Limits	109
	Creating New Complex Types by Extension	110
	Attribute Declarations	111
	Understanding Namespaces	112
	What Is a Namespace?	112
	URIs, URNs, and URLs	113
	Namespace Qualification in Instance Documents	114
	The W3C XML Schema-Related Namespaces	116
	Structuring Schemas	117
	Global Types and Local Elements versus Global Elements	117
	Named Types and Anonymous Types	118
	Modularity: The xs:include and xs:import Elements	118
	An Example of Importing Type Libraries	119
	Less Commonly Used W3C XML Schema Language Features	127

Is There Data or Not?	131
Reference	133
Resources	133
Chapter 5 Validating against Schemas	135
Requirements	135
Running the Utilities	136
Design	137
Java Implementation	139
Input Validation in XMLToCSVBasic.java	140
Output Validation in CSVToXMLBasic.java	142
C++ Implementation	144
Input Validation in XMLToCSVBasic.cpp	144
Output Validation in CSVToXMLBasic.cpp	145
Examples of Input Validation Failures	147
Resources	152
Chapter 6 Refining the Design	153
Why Refine the Design?	153
Making XML the Common Format	154
Analyzing the Legacy Non-XML Grammars	155
Describing the Legacy Non-XML Grammars	159
Representing the Legacy Non-XML Grammars in XML	162
Instance Document Design	162
File Description Document Design	163
Schemas for File Description Documents	165
Schemas for Source and Target Documents	168
Additional DOM Processing Considerations and Strategies	170
Multilingual Issues	174
Error Handling Strategy	175
High-Level Design	176
Source Converter Processing	176
Target Converter Processing	176
Summary of Classes	178
Detail Design	182
Main Routine Structures	182
Converter Base Class	184
SourceConverter Base Class (Extends Converter)	185
TargetConverter Base Class (Extends Converter)	188
RecordHandler Base Class	190
RecordReader Base Class (Extends RecordHandler)	195
RecordWriter Base Class (Extends RecordHandler)	200
DataCell Base Class	203

Java Implementation	205
C++ Implementation	207
References	208
Resources	208
Chapter 7 Converting CSV Files to and from XML, Revisited	209
CSV to XML: Functionality and Operation	210
Requirements	210
Running the Utility	211
Sample Input and Output: Invoice	213
XML to CSV: Functionality and Operation	218
Requirements	218
Running the Utility	219
Sample Input and Output: Purchase Order	220
Describing the File Formats	225
CSV Physical Characteristics	225
XML Output Characteristics	225
CSV File Grammar	226
Example File Description Documents	230
Schema Examples	233
High-Level Design Considerations	237
Grammar Analysis and Description	238
File Description Document Schemas	240
CSV to XML: Detail Design	249
Main Program	249
CSVSourceConverter Class (Extends SourceConverter)	250
CSVRecordReader Class (Extends RecordReader)	255
XML to CSV: Detail Design	261
Main Program	261
CSVTargetConverter Class (Extends TargetConverter)	262
CSVRecordWriter Class (Extends RecordWriter)	264
New DataCell Methods and Derived Classes	266
New DataCell Methods	266
DataCellAN Class	268
DataCellReal Class	268
DataCellDateMMsDDsYYYY Class	269
Java Implementation	271
C++ Implementation	273
Enhancements and Alternatives	275
Additional Data Types	275
Variety of Record Types	275
Efficiency and Performance	276
Resources	277

Chapter 8	Converting Flat Files to and from XML	279
Flat File to XML: Functionality and Operation		279
Requirements		279
Running the Utility		280
Sample Input and Output: Invoice		282
XML to Flat File: Functionality and Operation		291
Requirements		291
Running the Utility		291
Sample Input and Output: Purchase Order		293
Describing the File Formats		298
Flat File Physical Characteristics		300
XML Output Characteristics		300
Flat File Grammar		300
Example File Description Documents		313
Schema Examples		317
High-Level Design Considerations		321
Grammar Analysis and Description		321
File Description Document Schemas		323
Flat File to XML: Detail Design		325
Main Program		325
FlatSourceConverter Class (Extends SourceConverter)		326
FlatRecordReader Class (Extends RecordReader)		333
XML to Flat File: Detail Design		337
Main Program		337
FlatTargetConverter Class (Extends TargetConverter)		338
FlatRecordWriter Class (Extends RecordWriter)		342
New DataCell Methods and Derived Classes		345
New DataCell Methods		345
New Methods in DataCell Derived Classes		347
DataCellN Class		350
DataCellDateYYYYMMDD Class		352
Java Implementation		354
C++ Implementation		354
Enhancements and Alternatives		355
Additional Data Types		355
CSV Record Formats		355
Rounding versus Truncation		356
Group Fields		356
Redefined Fields		356
Resources		357

Chapter 9	Converting EDI to and from XML	359
	Overview of the X12 EDI Syntax and Standards	360
	X12 to XML: Functionality and Operation	363
	Requirements	363
	Running the Utility	364
	Sample Input and Output: 850 Purchase Order	367
	XML to X12: Functionality and Operation	374
	Requirements	374
	Running the Utility	374
	Sample Input and Output: 810 Invoice	376
	Describing the File Formats	384
	X12 File Physical Characteristics	385
	XML Output Characteristics	385
	Transaction Set Grammar	388
	Example File Description Documents	389
	Schema Examples	397
	Supplemental Data Store for Control Numbers	398
	High-Level Design Considerations	400
	Grammar Analysis and Description	400
	File Description Document Schemas	403
	X12 to XML: Detail Design	404
	Main Program	404
	X12SourceConverter Class (Extends SourceConverter)	405
	EDIRecordReader Class (Extends RecordReader)	412
	X12RecordReader Class (Extends EDIRecordReader)	425
	XML to X12: Detail Design	428
	Main Program	428
	X12TargetConverter Class (Extends TargetConverter)	429
	EDIRecordWriter Class (Extends RecordWriter)	433
	X12RecordWriter Class (Extends EDIRecordWriter)	438
	New DataCell Methods and Derived Classes	444
	DataCellX12N Class (Extends DataCellN)	444
	DataCellX12R Class (Extends DataCellReal)	445
	DataCellX12DT Class (Extends DataCellDateYYYYMMDD)	447
	DataCellX12TM Class	448
	Java Implementation	450
	C++ Implementation	452
	Enhancements and Alternatives	453
	Reference	454
	Resources	454

Chapter 10	Converting from One XML Format to Another with XSLT	457
	Why XSLT Is Important	457
	XSLT Overview	459
	A Simple Example: Hello World	460
	Another Simple Example: Changing Tag Names	462
	A General Approach to Using XSLT	465
	XPath Basics	467
	Structuring Stylesheets	471
	A Bit of Housekeeping	478
	The <code>xsl:output</code> Element	478
	Running Transformations from a Command Line	478
	Manipulating Content	479
	Adding and Removing Content	479
	Splitting Data Content	480
	Combining Data Content	481
	Changing an Attribute to an Element	484
	Changing an Element to an Attribute	485
	Solving Typical Navigation Problems	486
	Mapping a Flat Structure to a Hierarchy	486
	Mapping a Hierarchy to a Flat Structure	492
	Tips for Dealing with Other Navigation Problems	494
	Advanced Techniques for Processing Content	495
	Omitting Empty Elements and Attributes	495
	Converting Coded Values	497
	Handling Calculations	502
	Handling Namespaces	505
	Calling Non-XSLT Procedures	508
	References	511
	Resources	511
Chapter 11	Using the Conversion Techniques Together	513
	Pipe and Filter Revisited	513
	Sample Conversion Scenarios and Scripts	515
	Purchase Order: UBL to XML to CSV	515
	Invoice: Flat File to XML to EDI	517
	Campaign Contribution Reporting: CSV to XML to Flat File	519
	Building a System: Babel Blaster	520
	Version 1.0 Requirements	523
	Architectural Overview	524
	Trading Partner/Application Information	525
	Linking Pipes and Filters	525
	Version 1.1 Requirements	526
	Resources	527

Chapter 12	Building XML Support into a Business Application	529
	What Should Be “XMLized”?	529
	Devising an Architecture	531
	Selecting the XML Format	534
	Changing Your Code	537
	What about Databases?	542
	Other Approaches and APIs	542
	Non-XML Issues	546
	Resources	548
Chapter 13	Security, Transport, Packaging, and Other Issues	551
	Some General Observations about Security	551
	Dealing with Security	552
	Security Requirements and Exposure	553
	Countermeasures and Remediation Strategies	554
	Prevention Countermeasures	555
	Transport	557
	Packaging	558
	Common Combinations for Security, Transport, and Packaging	558
	Emerging Technologies	560
	What This Means for You	561
	Reference	562
Appendix A	GNU General Public License	563
Appendix B	Pseudocode Conventions	571
Appendix C	COM Essentials for the Non-COM Programmer	575
	Bibliography	581
	Credits	585
	Index	587

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.