Microsoft

Inside Microsoft Dynamics® AX 2012

The Microsoft Dynamics AX Team

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Control Number: 2012950241
ISBN: 978-0-7356-6710-5

Printed and bound in the United States of America.

Third Printing

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at mspinput@microsoft.com. Please tell us what you think of this book at *http://www.microsoft.com/learning/booksurvey*.

Acquisitions Editor: Anne Hamilton
Developmental Editor: Margaret Sherman with the Microsoft Dynamics AX Team
Project Editor: Valerie Woolley
Editorial Production: Christian Holdener, S4Carlisle Publishing Services
Technical Reviewer: Allan Iversen
Copyeditor: Andrew Jones
Indexer: Maureen Johnson, MoJo's Indexing Service
Cover: Twist Creative · Seattle

[LSI]

[2013-08-09]

Contents at a glance

	Foreword	xxiii
	Introduction	xxv
PART I	**A TOUR OF THE DEVELOPMENT ENVIRONMENT**	
CHAPTER 1	Architectural overview	3
CHAPTER 2	The MorphX development environment and tools	19
CHAPTER 3	Microsoft Dynamics AX and .NET	73
CHAPTER 4	The X++ programming language	87
PART II	**DEVELOPING WITH MICROSOFT DYNAMICS AX**	
CHAPTER 5	Designing the user experience	137
CHAPTER 6	The Microsoft Dynamics AX client	159
CHAPTER 7	Enterprise Portal	195
CHAPTER 8	Workflow in Microsoft Dynamics AX	245
CHAPTER 9	Reporting in Microsoft Dynamics AX	275
CHAPTER 10	BI and analytics	299
CHAPTER 11	Security, licensing, and configuration	351
CHAPTER 12	Microsoft Dynamics AX services and integration	385
CHAPTER 13	Performance	417
CHAPTER 14	Extending Microsoft Dynamics AX	493
CHAPTER 15	Testing	527
CHAPTER 16	Customizing and adding help	545
PART III	**UNDER THE HOOD**	
CHAPTER 17	The database layer	577
CHAPTER 18	The Batch framework	613
CHAPTER 19	Application domain frameworks	633

CHAPTER 20 **Reflection** **669**
CHAPTER 21 **Application models** **687**

Appendix: Resources for code upgrade *705*

Index *707*

Contents

Foreword xxiii

Introduction xxv

The history of Microsoft Dynamics AX xxvi

Who should read this book xxvii

Who should not read this book xxviii

Organization of this book xxviii

Conventions and features in this book xxix

System requirements xxix

Code samples xxx

Acknowledgments xxxi

Errata & book support xxxii

We want to hear from you xxxiii

Stay in touch xxxiii

PART I A TOUR OF THE DEVELOPMENT ENVIRONMENT

Chapter 1 Architectural overview 3

Introduction 3

Microsoft Dynamics AX five-layer solution architecture 4

Microsoft Dynamics AX application platform architecture 6

Application development environments 6

Data tier of the Microsoft Dynamics AX platform 7

Middle tier of the Microsoft Dynamics AX platform 7

Presentation tier of the Microsoft Dynamics AX platform 8

Microsoft Dynamics AX application meta-model architecture 9

Application data element types 10

MorphX user interface control element types 11

Workflow element types 12

Code element types 13
Services element types 13
Role-based security element types 14
Web client element types 14
Documentation and resource element types 16
License and configuration element types 16

Chapter 2 The MorphX development environment and tools 19

Introduction 19
Application Object Tree 20
Navigate through the AOT 21
Create elements in the AOT 23
Modify elements in the AOT 23
Refresh elements in the AOT 25
Element actions in the AOT 25
Element layers and models in the AOT 26
Projects 27
Create a project 27
Automatically generate a project 28
Project types 30
Property sheet 30
X++ code editor 31
Shortcut keys 32
Editor scripts 33
Label editor 33
Create a label 35
Reference labels from X++ 36
Code compiler 37
Best Practices tool 39
Rules 40
Suppress errors and warnings 41
Add custom rules 42

Debugger 43
Enable debugging 43
Debugger user interface 44
Debugger shortcut keys 47
Reverse Engineering tool 47
UML data model 48
UML object model 49
Entity relationship data model 51
Table Browser tool 52
Find tool 53
Compare tool 54
Start the Compare tool 55
Use the Compare tool 57
Compare APIs 58
Cross-Reference tool 60
Version control 62
Element life cycle 64
Common version control tasks 65
Work with labels 66
Synchronize elements 67
View the synchronization log 68
Show the history of an element 69
Compare revisions 70
View pending elements 70
Create a build 71
Integrate Microsoft Dynamics AX with other version control systems 71

Chapter 3 Microsoft Dynamics AX and .NET 73

Introduction 73
Use third-party assemblies 74
Use strong-named assemblies 74
Reference a managed DLL from Microsoft Dynamics AX 75

Code against the assembly in X++ 76
Write managed code 77
Debug managed code 81
Proxies 82
Hot swap assemblies on the server 84

Chapter 4 The X++ programming language 87

Introduction 87
Jobs 88
The type system 88
Value types 88
Reference types 89
Type hierarchies 89
Syntax 93
Variable declarations 93
Expressions 95
Statements 96
Macros 113
Comments 115
XML documentation 116
Classes and interfaces 117
Fields 118
Methods 118
Delegates 120
Pre- and post-event handlers 122
Attributes 123
Code access security 124
Compiling and running X++ as .NET CIL 126
Design and implementation patterns 128
Class-level patterns 129
Table-level patterns 131

PART II DEVELOPING WITH MICROSOFT DYNAMICS AX

Chapter 5 Designing the user experience 137

Introduction 137

A role-tailored design approach 139

User experience components 140

Navigation layer forms 141

Work layer forms 142

Role Center pages 142

Cues 143

Design Role Centers 143

Area pages 144

Design area pages 145

List pages 146

A simple scenario: taking a call from a customer 146

Use a list page as an alternative to a report 148

Design list pages 149

Details forms 150

Transaction details forms 153

Enterprise Portal web client user experience 155

Navigation layer forms 156

Work layer forms 157

Design for Enterprise Portal 157

Design for your users 157

Chapter 6 The Microsoft Dynamics AX client 159

Introduction 159

Working with forms 159

Form patterns 160

Form metadata 162

Form data sources 164
Form queries 170
Adding controls 172
Control overrides 173
Control data binding 173
Design node properties 173
Runtime modifications 174
Action controls 174
Layout controls 176
Input controls 178
ManagedHost control 179
Other controls 181
Using parts 181
Types of parts 181
Reference a part from a form 182
Adding navigation items 182
MenuItem 182
Menu 183
Menu definitions 183
Customizing forms with code 184
Method overrides 184
Auto variables 187
Business logic 188
Custom lookups 188
Integrating with the Microsoft Office client 189
Make data sources available to Office Add-ins 189
Build an Excel template 190
Build a Word template 191
Add templates for users 192

Chapter 7 Enterprise Portal 195

Introduction 195
Enterprise Portal architecture 196

Enterprise Portal components 198
Web parts 199
AOT elements 201
Datasets 201
Enterprise Portal framework controls 203
Developing for Enterprise Portal 216
Create a model-driven list page 217
Create a details page 219
AJAX 222
Session disposal and caching 223
Context 223
Data 225
Metadata 225
Proxy classes 226
ViewState 228
Labels 229
Formatting 230
Validation 231
Error handling 231
Security 232
Secure web elements 233
Record context and encryption 235
SharePoint integration 235
Site navigation 235
Site definitions, page templates, and web parts 237
Import and deploy a web part page 239
Enterprise Search 240
Themes 243

Chapter 8 Workflow in Microsoft Dynamics AX 245

Introduction 245
Microsoft Dynamics AX 2012 workflow infrastructure 246
Windows Workflow Foundation 249

Key workflow concepts 250
Workflow document and workflow document class 250
Workflow categories 251
Workflow types 251
Event handlers 252
Menu items 252
Workflow elements 252
Queues 253
Providers 254
Workflows 255
Workflow instances 256
Work items 256
Workflow architecture 256
Workflow runtime 257
Workflow runtime interaction 258
Logical approval and task workflows 260
Workflow life cycle 262
Implementing workflows 263
Create workflow artifacts, dependent artifacts, and business logic 264
State management 266
Create a workflow category 268
Create the workflow document class 268
Add a workflow display menu item 270
Activate the workflow 270

Chapter 9 Reporting in Microsoft Dynamics AX 275

Introduction 275
Inside the Microsoft Dynamics AX 2012 reporting framework 276
Client-side reporting solutions 276
Server-side reporting solutions 277
Report execution sequence 278
Plan your reporting solution 279

Reporting and users 279
Roles in report development 280
Create production reports 281
Model elements for reports 282
SSRS extensions 285
Microsoft Dynamics AX extensions 286
Create charts for Enterprise Portal 289
Microsoft Dynamics AX chart development tools 289
Integration with Microsoft Dynamics AX 290
Data series 292
Add interactive functions to a chart 294
Override the default chart format 296
Troubleshoot the reporting framework 296
The report server cannot be validated 297
A report cannot be generated 297
A chart cannot be debugged because of SharePoint sandbox issues 297

Chapter 10 BI and analytics 299

Introduction 299
Components of the Microsoft Dynamics AX 2012 BI solution 299
Implementing the prebuilt BI solution 301
Implement the prerequisites 302
Configure an SSAS server 302
Deploy cubes 303
Deploy cubes in an environment with multiple partitions 305
Process cubes 307
Provision users in Microsoft Dynamics AX 308
Customizing the prebuilt BI solution 309
Configure analytic content 310
Customize cubes 311
Extend cubes 319
Creating cubes 323

Identify requirements 324
Define metadata. 325
Generate and deploy the cube 328
Add KPIs and calculations. 333
Displaying analytic content in Role Centers. 333
Provide insights tailored to a persona 334
Choose a presentation tool based on a persona 335
SQL Server Power View reports. 335
Excel 340
Business Overview web part and KPI List web part 341
Develop reports with Report Builder 346
Develop analytic reports by using Visual Studio tools for Microsoft Dynamics AX 346

Chapter 11 Security, licensing, and configuration 351

Introduction 351
Security framework overview 351
Authentication 352
Authorization 353
Data security 356
Develop security artifacts 356
Set permissions for a form 356
Set permissions for server methods 359
Set permissions for controls 359
Create privileges. 359
Assign privileges and duties to security roles 361
Use valid time state tables 362
Validate security artifacts 363
Create users. 363
Assign users to roles 363
Set up segregation of duties rules 364
Create extensible data security policies 364
Data security policy concepts 365

Develop an extensible data security policy 365
Debug extensible data security policies 368
Security coding 369
Table permissions framework 369
Code access security 371
Best practice rules 372
Security debugging 373
Licensing and configuration 376
Configuration hierarchy 378
Configuration keys 378
Use configuration keys 380
Types of CALs 381
Customization and licensing 383

Chapter 12 Microsoft Dynamics AX services and integration 385

Introduction 385
Types of Microsoft Dynamics AX services 387
System services 387
Custom services 388
Document services 392
Security considerations 400
Publish Microsoft Dynamics AX services 400
Consume Microsoft Dynamics AX services 401
Sample WCF client for *CustCustomerService* 402
Consume system services 404
Update business documents 407
Invoke custom services asynchronously 409
The Microsoft Dynamics AX send framework 411
Implementing a trigger for transmission 411
Configure transmission mechanisms 414
Consume external web services from Microsoft Dynamics AX 414
Performance considerations 415

Chapter 13 Performance **417**

Introduction 417

Client/server performance 417

Reduce round-trips between the client and the server. 418

Write tier-aware code 422

Transaction performance. 426

Set-based data manipulation operators 427

Restartable jobs and optimistic concurrency 444

Caching 446

Field lists 456

Field justification 462

Performance configuration options. 462

SQL Administration form 462

Server Configuration form 463

AOS configuration 463

Client configuration 464

Client performance 465

Number sequence caching. 465

Extensive logging. 465

Master scheduling and inventory closing 465

Coding patterns for performance 465

Execute X++ code as CIL. 466

Use parallel execution effectively 466

The SysOperation framework. 467

Patterns for checking to see whether a record exists 472

Run a query only as often as necessary 473

When to prefer two queries over a join 474

Indexing tips and tricks 475

When to use *firstfast* 476

Optimize list pages 476

Aggregate fields to reduce loop iterations 477

Performance monitoring tools 478

Microsoft Dynamics AX Trace Parser 479

Monitor database activity 488
Use the SQL Server connection context to find the SPID or user behind a client session 489
The client access log 490
Visual Studio Profiler 490

Chapter 14 Extending Microsoft Dynamics AX 493

Introduction 493
The SysOperation framework 493
SysOperation framework classes 494
SysOperation framework attributes 495
Comparing the SysOperation and RunBase frameworks 495
RunBase example: *SysOpSampleBasicRunbaseBatch* 496
SysOperation example: *SysOpSampleBasicController* 504
The RunBase framework 510
Inheritance in the RunBase framework 510
Property method pattern 511
Pack-unpack pattern 512
Client/server considerations 516
The extension framework 516
Create an extension 517
Extension example 518
Eventing 520
Delegates 521
Pre and *post* events 522
Event handlers 523
Eventing example 524

Chapter 15 Testing 527

Introduction 527
New unit testing features in Microsoft Dynamics AX 2012 527
Use predefined test attributes 528

Create test attributes and filters 530
Microsoft Visual Studio 2010 test tools. 533
Use all aspects of the ALM solution 534
Use an acceptance test driven development approach 535
Use shared steps. 536
Record shared steps for fast forwarding 537
Develop test cases in an evolutionary manner 538
Use ordered test suites for long scenarios. 539
Putting everything together. 540
Execute tests as part of the build process 540
Use the right tests for the job 542

Chapter 16 Customizing and adding help 545

Introduction 545
Help system overview 546
Microsoft Dynamics AX client 547
Help viewer 547
Help server 548
AOS. 549
Help content overview. 549
Topics 549
Publisher 550
Table of contents 550
Summary page 550
Create content. 550
Create a topic in HTML 552
Add labels, fields, and menu items to a topic. 559
Make a topic context-sensitive 561
Update content from other publishers. 562
Create a table of contents file 563
Create non-HTML content 565
Publish content 567
Add a publisher to the *Web.config* file 569

Publish content to the Help server 570
Set Help document set properties 571
Troubleshoot the Help system 572
The Help viewer cannot display content 572
The Help viewer cannot display the table of contents 573

PART III UNDER THE HOOD

Chapter 17 The database layer 577

Introduction 577
Temporary tables 577
InMemory temporary tables 578
TempDB temporary tables 582
Creating temporary tables 583
Surrogate keys 585
Alternate keys 587
Table relations 588
EDT relations and table relations 588
Foreign key relations 590
The *CreateNavigationPropertyMethods* property 591
Table inheritance 594
Modeling table inheritance 594
Table inheritance storage model 596
Polymorphic behavior 596
Performance considerations 598
Unit of Work 599
Date-effective framework 601
Relational modeling of date-effective entities 601
Support for data retrieval 603
Run-time support for data consistency 604
Full-text support 606

The QueryFilter API 607
Data partitions 610
Partition management 611
Development experience 611
Run-time experience 611

Chapter 18 The batch framework 613

Introduction 613
Batch processing in Microsoft Dynamics AX 2012 613
Common uses of the batch framework 614
Performance 615
Create and execute a batch job 615
Create a batch-executable class 616
Create a batch job 618
Use the Batch API 623
Manage batch execution 625
Configure the batch server 625
Create a batch group 626
Manage batch jobs 628
Debug a batch task 629
Configure AOS for batch debugging 629
Configure Visual Studio for debugging X++ in a batch 630

Chapter 19 Application domain frameworks 633

Introduction 633
The organization model framework 634
How the organization model framework works 634
When to use the organization model framework 636
The product model framework 643
How the product model framework works 643
When to use the product model framework 647

Extending the product model framework 647
The operations resource framework 648
How the operations resource framework works 648
When to use the operations resource framework 652
Extensions to the operations resource framework 652
MorphX model element prefixes for the operations resource framework 654
The dimension framework 654
How the dimension framework works 654
Constrain combinations of values 656
Create values 656
Extend the dimension framework 657
Query data 658
Physical table references 659
The accounting framework 659
How the accounting framework works 660
When to use the accounting framework 662
Extensions to the accounting framework 662
Accounting framework process states 662
MorphX model element prefixes for the accounting framework 663
The source document framework 664
How the source document framework works 664
When to use the source document framework 665
Extensions to the source document framework 666
MorphX model element prefixes for the source document framework 667

Chapter 20 Reflection 669

Introduction 669
Reflection system functions 670
Intrinsic functions 670
typeOf system function 671

classIdGet system function 672
Reflection APIs 673
Table data API 673
Dictionary API 676
Treenodes API 680
TreeNodeType 683

Chapter 21 Application models **687**

Introduction 687
Layers 688
Models 690
Element IDs 692
Create a model 693
Prepare a model for publication 694
Set the model manifest 694
Export the model 695
Sign the model 696
Import model files 697
Upgrade a model 699
Move a model from test to production 700
Create a test environment 701
Prepare the test environment 701
Deploy the model to production 701
Element ID considerations 702
Model store API 703

Appendix: Resources for code upgrade *705*
Index *707*

Foreword

Microsoft Dynamics AX and its predecessor, Axapta, have always benefited from a very active and enthusiastic developer community. Some of those developers are employed by mid-size to large firms that build their business selling solutions built on Microsoft Dynamics AX. Others are in the IT departments of companies using Microsoft Dynamics AX as mission-critical infrastructure.

One of the consistent pieces of feedback I've received from those developers over the years is how the raw power and agility provided by the Microsoft Dynamics AX toolset and metadata environment make them more productive than any other line of business application framework. With Microsoft Dynamics AX 2012, we have taken the productivity and power of that toolset to a whole new level; delivering event-based customization, delta customization of forms, a new editor, date effectivity, and subtype/supertype support, to name just a few.

We continued the journey to expose the power of Microsoft SQL Server Reporting Services (SSRS) and Analysis Services directly within Microsoft Dynamics AX, moving all of the out-of-the-box reports and business intelligence inside the platform.

We back all of that up with almost three times the application footprint of prior versions of Microsoft Dynamics AX, truly making Microsoft Dynamics AX both a powerful developer environment and a rich out-of-the-box suite of applications.

This book focuses on the enhancements in the Microsoft Dynamics AX 2012 toolset and is written by the team that brought you those tools. It's truly an insider's view of the entire Microsoft Dynamics AX development and run-time environment. I hope you enjoy it as much as we enjoyed writing the book and creating the product.

Thanks,

Hal Howard

Head of Product Development, Microsoft Dynamics AX

Corporate Vice President, Microsoft Dynamics Research and Development

Introduction

Microsoft Dynamics AX 2012 represents a new generation of enterprise resource planning (ERP) software. With over 1,000 new features and prebuilt industry capabilities for manufacturing, distribution, services, retail, and public sector, Microsoft Dynamics AX 2012 provides a robust platform for developers to deliver specialized functionality more efficiently to the industries that they support.

Microsoft Dynamics AX 2012 is a truly global solution, able to scale with any business as it grows. It is simple enough to deploy for a single business unit in a single country/region, yet robust enough to support the unique requirements for business systems in 36 countries/regions—all from a single-instance deployment of the software.

For this version of Microsoft Dynamics AX, the entire codebase was analyzed and, where necessary, reengineered, so that the application is built more holistically around a set of unified principles. As Microsoft Technical Fellow Mike Ehrenberg explains:

> The heart of Microsoft Dynamics AX 2012 is a set of unified, natural models that let you see, measure, and change your business. In developing this release, every application concept involved in representing the business in software was reexamined. In each case, limitations that forced workarounds and compromises in older ERP products were lifted, and new capabilities were added to provide an even richer software representation of a business and its structure, processes, and policies. Unified, natural Microsoft Dynamics AX 2012 models make modeling simple businesses fast and easy and yet still provide the richness and flexibility to represent the most complex organizations.

Early adopters have also weighed in on the benefits of Microsoft Dynamics AX 2012:

> Microsoft Dynamics AX 2012 allows us to collaborate within our organization and with our constituents . . . using built-in controls and fund/encumbrance accounting capabilities to ensure compliance with Public Sector requirements . . . and using out-of the-box Business Analytics and Intelligence . . . so executives can make effective decisions in real time.
>
> Mike Bailey
>
> Director of Finance and Information Services
>
> City of Redmond (WA)

> With the latest release, developing for and customizing Microsoft Dynamics AX will be easier than ever. Developers will be able to work with X++ directly from within Microsoft Visual Studio and enjoy more sophisticated features in the X++ editor, for example. Also, the release includes more prebuilt interoperability with Microsoft SharePoint Server and SQL Server Reporting Services, so that developers spend less time on mundane work when setting up customer systems.
>
> Guido Van de Velde
>
> Director of MECOMS™
>
> Ferranti Computer Systems

Microsoft Dynamics AX 2012 is substantially different from its predecessor, which can mean a steep learning curve for developers and system implementers who have worked with previous versions. However, by providing a broad overview of the architectural changes, new technologies, and tools for this release, the authors of *Inside Microsoft Dynamics AX 2012* have created a resource that will help reduce the time that it takes for developers to become productive with this version of Microsoft Dynamics AX.

The history of Microsoft Dynamics AX

Historically, Microsoft Dynamics AX encompasses more than 25 years of experience in business application innovation and developer productivity. Microsoft acquired the predecessor of Microsoft Dynamics AX, called Axapta, in 2002, with its purchase of the Danish company Navision A/S. The success of the product has spurred an increasing commitment of research and development resources, which allows Microsoft Dynamics AX to grow and strengthen its offering continuously.

The development team that created Microsoft Dynamics AX 2012 consists of three large teams, two that are based in the United States (Fargo, North Dakota, and Redmond, Washington) and one that is based in Denmark (Copenhagen). The Fargo team focuses on finance and human resources (HR), the Redmond team concentrates on project management and accounting and customer relationship management (CRM), and the Copenhagen team delivers supply chain management (SCM). In addition, a framework team develops infrastructure components, and a worldwide distributed team localizes the Microsoft Dynamics AX features to meet national regulations or local differences in business practices in numerous languages and markets around the world.

To clarify a few aspects of the origins of Microsoft Dynamics AX, the authors contacted people who participated in the early stages of the Microsoft Dynamics AX development cycle. The first question we asked was, "How was the idea of using X++ as the programming language for Microsoft Dynamics AX conceived?"

> We had been working with an upgraded version of XAL for a while called OO XAL back in 1996/1997. At some point in time, we stopped and reviewed our approach and looked at other new languages like Java. After working one long night, I decided that our approach had to change to align with the latest trends in programming languages, and we started with X++.
>
> Erik Damgaard
>
> Cofounder of Damgaard Data

Of course, the developers had several perspectives on this breakthrough event.

> One morning when we came to work, nothing was working. Later in the morning, we realized that we had changed programming languages! But we did not have any tools, so for months we were programming in Notepad without compiler or editor support.
>
> Anonymous developer

Many hypotheses exist regarding the origin of the original product name, Axapta. Axapta was a constructed name, and the only requirement was that the letter X be included, to mark the association with its predecessor, XAL. The X association carries over in the name Microsoft Dynamics AX.

Who should read this book

This book explores the technology and development tools in Microsoft Dynamics AX 2012. It is designed to help new and existing Microsoft Dynamics AX developers by providing holistic and in-depth information about developing for Microsoft Dynamics AX 2012—information that may not be available from other resources, such as SDK documentation, blogs, or forums. It aids developers who are either customizing Microsoft Dynamics AX 2012 for a specific implementation or building modules or applications that blend seamlessly with Microsoft Dynamics AX 2012. System implementers and consultants will also find much of the information useful.

Assumptions

To get full value from this book, you should have knowledge of common object-oriented concepts from languages such as C++, C#, and Java. You should also have knowledge of relational database concepts. Knowledge of Structured Query Language (SQL) and Microsoft .NET technology are also advantageous. Transact-SQL statements are used to perform relational database tasks, such as data updates and data retrieval.

Who should not read this book

This book is not aimed at those who install, upgrade, or deploy Microsoft Dynamics AX 2012. It is also beyond the scope of this book to include details about the sizing of production environments. For more information about these topics, refer to the extensive installation and implementation documentation that is supplied with the product or available on TechNet, MSDN, and other websites.

The book also does not provide instructions for those who configure parameter options within Microsoft Dynamics AX 2012 or the business users who use the application in their day-to-day work. For assistance with these activities, refer to the help that is shipped with the product and available on TechNet at *http://technet.microsoft.com/en-us/library/gg852966.aspx*.

Organization of this book

Although *Inside Microsoft Dynamics AX 2012* does not provide exhaustive coverage of every feature in Microsoft Dynamics AX 2012, it does offer a broad view that will benefit developers as they develop for the product.

This book is divided into three sections, each of which focuses on Microsoft Dynamics AX 2012 from a different angle. Part I, "A tour of the development environment," provides an overview of the Microsoft Dynamics AX 2012 architecture that has been written with developers in mind. The chapters in Part I also provide a tour of the internal Microsoft Dynamics AX 2012 development environment to help new developers familiarize themselves with the designers and tools that they will use to implement their customizations, extensions, and integrations.

Part II, "Developing with Microsoft Dynamics AX 2012," provides the information that developers need in order to customize and extend Microsoft Dynamics AX 2012. In addition to explanations of the features, many chapters include examples, some of which are available as downloadable files that can help you learn how to code for

Microsoft Dynamics AX. For information about how to access these files, see the "Code samples" section, later in this introduction.

Part III, "Under the hood," is largely devoted to illustrating how developers can use the underlying foundation of the Microsoft Dynamics AX 2012 application frameworks to develop their solutions, with a focus on the database layer, system and application frameworks, reflection, and models.

Conventions and features in this book

This book presents information using the following conventions, which are designed to make the information readable and easy to follow.

- Application Object Tree (AOT) paths use backslashes to separate nodes, such as Forms\AccountingDistribution\Methods.
- The names of methods, functions, properties and property values, fields, and nodes appear in italics.
- Registry keys and T-SQL commands appear in capital letters.
- User interface (UI) paths use angle brackets to indicate actions—for example, "On the File menu, point to Tools > Options."
- Boxed elements with labels such as "Note" provide additional information or alternative methods for completing a step successfully.
- Text that you type (apart from code blocks) appears in bold.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, "Press Alt+Tab" means that you hold down the Alt key while you press the Tab key.

System requirements

To work with sample code, you must have the RTM version of Microsoft Dynamics AX 2012 installed. For information about the system requirements for installing Microsoft Dynamics AX 2012, see the Microsoft Dynamics AX 2012 Installation Guide at *http://www.microsoft.com/en-us/download/details.aspx?id=12687*.

You must also have an Internet connection to download the sample files that are provided as supplements to many of the chapters.

Note Some of the features described in this book, such as data partitioning and the EP Chart Control, apply only to the Microsoft Dynamics AX 2012 R2. That is noted where those features are discussed.

Code samples

Most of the chapters in this book include code examples that let you interactively try out the new material presented in the main text. You can download the example code from the following page:

http://go.microsoft.com/FWLink/?Linkid=263524

Follow the instructions to download the InsideDynaAX2012_667105_CompanionContent.zip file.

Installing the code samples

Follow these steps to install the code samples on your computer:

1. Unzip the InsideDynaAX2012_667105_CompanionContent.zip file that you downloaded from the book's website.
2. If prompted, review the displayed end user license agreement. If you accept the terms, select the accept option, and then click Next.

Note If the license agreement doesn't appear, you can access it from the same webpage from which you downloaded the InsideDynaAX2012_667105_CompanionContent.zip file.

Using the code samples

The code examples referenced in each chapter are provided as both .xpo files that you can import into Microsoft Dynamics AX and Visual Studio projects that you can open through the corresponding .csproj files. Many of these examples are incomplete, and you cannot import and run them successfully without following the steps indicated in the associated chapter.

Acknowledgments

We want to thank all the people who assisted us in bringing this book to press. We apologize for anyone whose name we missed.

Microsoft Dynamics product team

Special thanks go to the following colleagues, whom we're fortunate to work with.

Margaret Sherman, who pitched the book to Microsoft Press, provided us with training in how to use templates and style sheets, created a schedule for writing, prodded us when we needed prodding to keep the writing process moving along, and provided editorial feedback on every chapter. Thank you, Margaret. This book absolutely would not have seen the light of day without you!

Mark Baker and Steve Kubis, who contributed ace project management and editing work.

Hal Howard, Richard Barnwell, and Ann Beebe, who sponsored the project and provided resources for it.

We're also grateful to the following members of the product team, who provided us with the reviews and research that helped us refine this book:

Ned Baker
Ian Beck
Andy Blehm
Jim Brotherton
Ed Budrys
Gregory Christiaens
Ahmad El Husseini
Josh Honeyman
Hitesh Jawa
Vijeta Johri
Bo Kampmann
Vinod Kumar
Josh Lange
Mey Meenakshisundaram
Igor Menshutkin
Jatan Modi
Sasha Nazarov
Adrian Orth
Christopher Read (Entirenet)
Bruce Rivard
Gana Sadasivam
Alex Samoylenko
Karen Scipi
Ramesh Shankar
Tao Wang
Lance Wheelwright
Chunke Yang
Arif Kureshy

In addition, we want to thank Joris de Gruyter of Streamline Systems LLC. His SysTestListenerTRX code samples on CodePlex (*http://dynamicsaxbuild.codeplex.com/releases*), with supporting documentation on his blog (*http://daxmusings.blogspot.com/*), and his collaboration as we investigated this approach for executing SysTests from Microsoft Dynamics AX were valuable resources as we prepared the chapter on testing.

Microsoft Press

Another big thank-you goes to the great people at Microsoft Press for their support and expertise throughout the writing and publishing process.

Valerie Woolley, the Content Project Manager for the book, who provided ongoing support and guidance throughout the life of the project.

Anne Hamilton—Acquisitions Editor

Christian Holdener—Production Project Manager with S4Carlisle Publishing Services

Allan Iversen—Technical Reviewer

Andrew Jones—Copy Editor

New arrivals

Finally, we would like to welcome the following youngest members of the Microsoft extended family, the children and grandchildren of the authors who arrived during the months that we were laboring on this book:

Charlie Hendrix Bird
Grace Elizabeth Marie Garty
Gavin Roy Healy
Kayden John Healy
Amrita Nalla
Nilay Pandya

Errata & book support

We've made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site at oreilly.com:

http://go.microsoft.com/FWLink/?Linkid=263523

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *mspinput@microsoft.com*.

Please note that product support for Microsoft software is not offered through the addresses above.

We want to hear from you

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at

http://www.microsoft.com/learning/booksurvey

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

Stay in touch

Let's keep the conversation going! We're on Twitter: *http://twitter.com/MicrosoftPress*

CHAPTER 3

Microsoft Dynamics AX and .NET

In this chapter

Introduction 73
Use third-party assemblies 74
Write managed code 77
Hot swap assemblies on the server 84

Introduction

Complex systems, such as Microsoft Dynamics AX 2012, are often deployed in heterogeneous environments that contain several disparate systems. Often, these systems contain legacy data that might be required for running Microsoft Dynamics AX, or they might offer functionality that is vital for running the organization.

Microsoft Dynamics AX 2012 offers several ways of integrating with other systems. For example, your organization might need to harvest information from old Microsoft Excel files. To do this, you could write a simple add-on in Microsoft Visual Studio and easily integrate it with Microsoft Dynamics AX. Or your organization might have a legacy system that is physically located in a distant location that requires invoice information to be sent to it in a fail-safe manner. In this case, you could set up a message queue to perform the transfers. You could use the Microsoft .NET Framework to interact with the message queue from within Microsoft Dynamics AX.

This chapter focuses on some of the ways that you can integrate Microsoft Dynamics AX with other systems by taking advantage of managed code through X++ code. One way is to consume managed code directly from X++ code; another way is to author or extend existing business logic in managed code by using the Visual Studio environment. To facilitate this interoperability, Microsoft Dynamics AX provides the managed code with managed classes (called *proxies*) that represent X++ artifacts. This allows you to write managed code that uses the functionality these proxies provide in a type-safe and convenient manner.

In both cases, the .NET Framework provides access to the functionality, and this functionality is used in Microsoft Dynamics AX.

Note You can also make Microsoft Dynamics AX functionality available to other systems by using services. For more information, see Chapter 12, "Microsoft Dynamics AX services and integration."

Use third-party assemblies

Sometimes, you can implement the functionality that you are looking to provide by using a managed component (a .NET assembly) that you purchase from a third-party vendor. Using these dynamic-link libraries (DLLs) can be—and often is—more cost effective than writing the code yourself. These components are wrapped in managed assemblies in the form of .dll files, along with their Program Database (PDB) files, which contain symbol information that is used in debugging, and their XML files, which contain documentation that is used for IntelliSense in Visual Studio. Typically, these assemblies come with an installation program that often installs the assemblies in the global assembly cache (GAC) on the computer that consumes the functionality. This computer can be either on the client tier, on the server tier, or both. Only assemblies with strong names can be installed in the GAC.

Use strong-named assemblies

It is always a good idea to use a DLL that has a strong name, which means that the DLL is signed by the author, regardless of whether the assembly is stored in the GAC. This is true for assemblies that are installed on both the client tier and the server tier. A strong name defines the assembly's identity by its simple text name, version number, and culture information (if provided)—plus a public key and a digital signature. Assemblies with the same strong name are expected to be identical.

Strong names satisfy the following requirements:

- **Guarantee name uniqueness by relying on unique key pairs.** No one can generate the same assembly name that you can, because an assembly generated with one private key has a different name than an assembly generated with another private key.
- **Protect the version lineage of an assembly.** A strong name can ensure that no one can produce a subsequent version of your assembly. Users can be sure that the version of the assembly that they are loading comes from the same publisher that created the version the application was built with.
- **Provide a strong integrity check.** Passing the .NET Framework security checks guarantees that the contents of the assembly have not been changed since it was built. Note, however, that by themselves, strong names do not imply a level of trust such as that provided by a digital signature and supporting certificate.

When you reference a strong-named assembly, you can expect certain benefits, such as versioning and naming protection. If the strong-named assembly references an assembly with a simple name, which does not have these benefits, you lose the benefits that you derive by using a strong-named assembly and open the door to possible DLL conflicts. Therefore, strong-named assemblies can reference only other strong-named assemblies.

If the assembly that you are consuming does not have a strong name, and is therefore not installed in the GAC, you must manually copy the assembly (and the assemblies it depends on, if applicable) to a directory where the .NET Framework can find it when it needs to load the assembly for execution.

It is a good practice to place the assembly in the same directory as the executable that will ultimately load it (in other words, the folder on the client or the server in which the application is located). You might also want to store the assembly in the Client\Bin directory (even if it is used on the server exclusively), so that the client can pick it up and use it for IntelliSense.

Reference a managed DLL from Microsoft Dynamics AX

Microsoft Dynamics AX 2012 does not have a built-in mechanism for bulk deployment or installation of a particular DLL on client or server computers, because each third-party DLL has its own installation process. You must do this manually by using the installation script that the vendor provides or by placing the assemblies in the appropriate folders.

After you install the assembly on the client or server computer, you must add a reference to the assembly in Microsoft Dynamics AX, so that you can program against it in X++. You do this by adding the assembly to the *References* node in the Application Object Tree (AOT).

To do this, right-click the *References* node, and then click Add Reference. A dialog box like the one shown in Figure 3-1 appears.

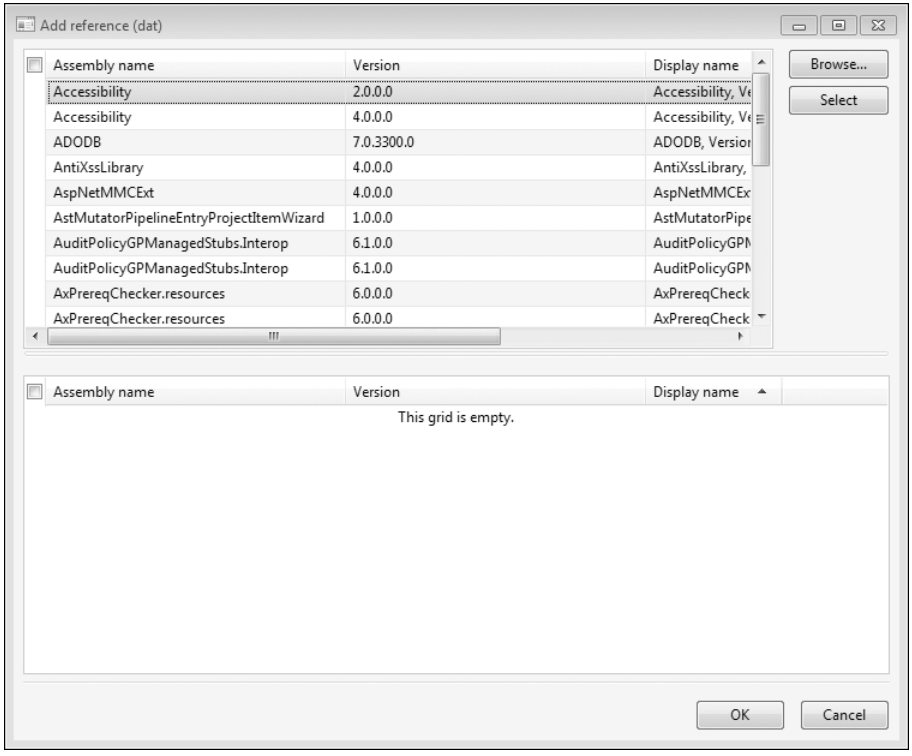

FIGURE 3-1 Adding a reference to a third-party assembly.

The top pane of the dialog box shows the assemblies that are installed in the GAC. If your assembly is installed in the GAC, click Select to add the reference to the *References* node. If the assembly is located in either the Client\Bin or the Server\Bin binary directory, click Browse. A file browser dialog box will appear where you can select your assembly. After you choose your assembly, it will appear in the bottom pane and will be added when you click OK.

Code against the assembly in X++

After you add the assembly, you are ready to use it from X++. If you install the code in the Client\Bin directory, IntelliSense features are available to help you edit the code. You can now use the managed code features of X++ to instantiate public managed classes, call methods on them, and so on. For more information, see Chapter 4, "The X++ programming language."

Note that there are some limitations to what you can achieve in X++ when calling managed code. One such limitation is that you cannot easily code against generic types (or execute generic methods). Another stems from the way the X++ interpreter works. Any managed object is represented as an instance of type *ClrObject*, and this has some surprising manifestations. For instance, consider the following code:

```
static void TestClr(Args _args)
{
    if (System.Int32::Parse("0"))
    {
        print "Do not expect to get here";
    }
    pause;
}
```

Obviously, you wouldn't expect the code in the *if* statement to execute because the result of the managed call is *0*, which is interpreted as *false*. However, the code actually prints the string literal because the return value of the call is a *ClrObject* instance that is not null (in other words, *true*). You can solve these problems by storing results in variables before use: the assignment operator will correctly unpack the value, as shown in the following example:

```
static void TestClr(Args _args)
{
    int i = System.Int32::Parse("0");
    if (i)
    {
        print "Do not expect to get here";
    }
    pause;
}
```

Write managed code

Sometimes your requirements cannot be satisfied by using an existing component and you have to roll up your sleeves and develop some code—in either C# or VB.NET. Microsoft Dynamics AX has great provisions for this: the integration features between Microsoft Dynamics AX and Visual Studio give you the luxury of dealing with X++ artifacts (classes, tables, and enumerations) as managed classes that behave the way that a developer of managed code would expect. The Microsoft Dynamics AX Business Connector manages the interaction between the two environments. Broadly speaking, you can create a project in Visual Studio as you normally would, and then add that project to the Visual Studio *Projects* node in the AOT. This section walks you through the process.

This example shows how to create managed code in C# (VB.NET could also be used) that reads the contents of an Excel spreadsheet and inserts the contents into a table in Microsoft Dynamics AX. This example is chosen to illustrate the concepts described in this chapter rather than for the functionality it provides.

Note The example in this section requires the Microsoft.ACE.OLEDB.12.0 provider to read data from Excel. You can download the provider from *http://www.microsoft.com/en-us/download/confirmation.aspx?id=23734*.

The process is simple: you author the code in Visual Studio, and then add the solution to Application Explorer, which is just the name for the AOT in Visual Studio. Then, functionality from Microsoft Dynamics AX is made available for consumption by the C# code, which illustrates the proxy feature.

Assume that the Excel file contains the names of customers and the date that they registered as customers with your organization, as shown in Figure 3-2.

	A	B	C	D	E	F
1	Name	Date				
2	Abercrombie, Kim	04/28/74				
3	Abolrous, Hazem	06/22/49				
4	Abrus, Luka	03/19/47				
5	Abu-Dayah, Ahmad	08/11/65				
6	Acevedo, Humberto	04/17/85				
7	Achong, Gustavo	09/09/80				
8	Ackerman, Pilar	09/08/81				
9	Adalsteinsson, Gudmundur	04/17/86				
10	Adams, Terry	06/22/49				
11	Affronti, Michael					

Sheet1 Sheet2 Sheet3

FIGURE 3-2 Excel spreadsheet that contains a customer list.

Also assume that you've defined a table (called, say, CustomersFromExcel) in the AOT that will end up containing the information, subject to further processing. You could go about reading the information from the Excel files from X++ in several ways: one is by using the Excel automation model; another is by manipulating the Office Open XML document by using the XML classes.

However, because it is so easy to read the contents of Excel files by using ADO.NET, that is what you decide to do. You start Visual Studio, create a C# class library called *ReadFromExcel*, and then write the following code:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Contoso
{
    using System.Data;
    using System.Data.OleDb;
    public class ExcelReader
    {
        static public void ReadDataFromExcel(string filename)
        {
            string connectionString;
            OleDbDataAdapter adapter;
            connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;"
            + "Data Source=" + filename + ";"
            + "Extended Properties='Excel 12.0 Xml;"

            + "HDR=YES'"; // Since sheet has row with column titles

            adapter = new OleDbDataAdapter(
                "SELECT * FROM [sheet1$]",
                connectionString);
            DataSet ds = new DataSet();
            // Get the data from the spreadsheet:
            adapter.Fill(ds, "Customers");
            DataTable table = ds.Tables["Customers"];
            foreach (DataRow row in table.Rows)
            {
                string name = row["Name"] as string;
                DateTime d = (DateTime)row["Date"];
            }
        }
    }
}
```

The *ReadDataFromExcel* method reads the data from the Excel file given as a parameter, but it does not currently do anything with that data. You still need to establish a connection to the Microsoft Dynamics AX system to store the values in the table. There are several ways of doing this, but in this case, you will simply use the Microsoft Dynamics AX table from the C# code by using the proxy feature.

The first step is to make the Visual Studio project (that contains the code) into a Microsoft Dynamics AX "citizen." You do this by selecting the Add ReadFromExcel To AOT menu item on the Visual Studio project. When this is done, the project is stored in the AOT and can use all of the functionality that is available for nodes in the AOT. The project can be stored in separate layers, be imported and exported, and so on. The project is stored in its entirety, and you can open Visual Studio to edit the project by clicking Edit on the context menu, as shown in Figure 3-3.

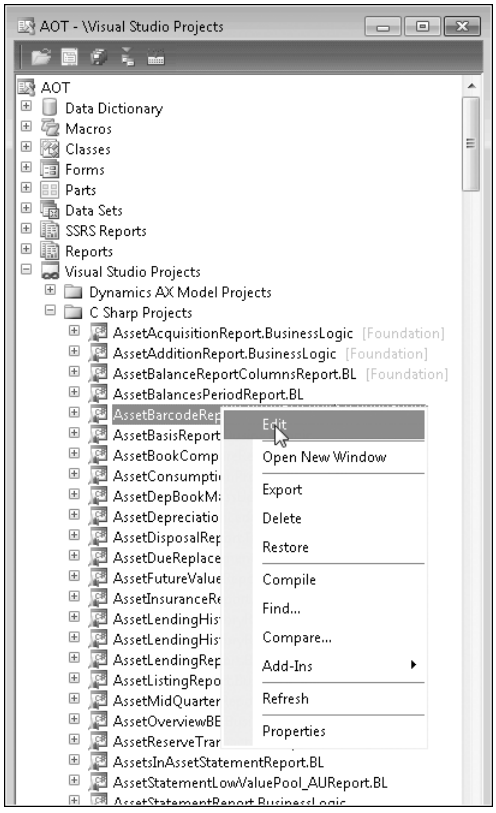

FIGURE 3-3 Context menu for Visual Studio projects that are stored in the AOT.

Tip You can tell that a project has been added to the AOT because the Visual Studio project icon is updated with a small Microsoft Dynamics AX icon in the lower-left corner.

With that step out of the way, you can use the version of the AOT that is available in Application Explorer in Visual Studio to fetch the table to use in the C# code (see Figure 3-4). If the Application Explorer window is not already open, you can open it by clicking Application Explorer on the View menu.

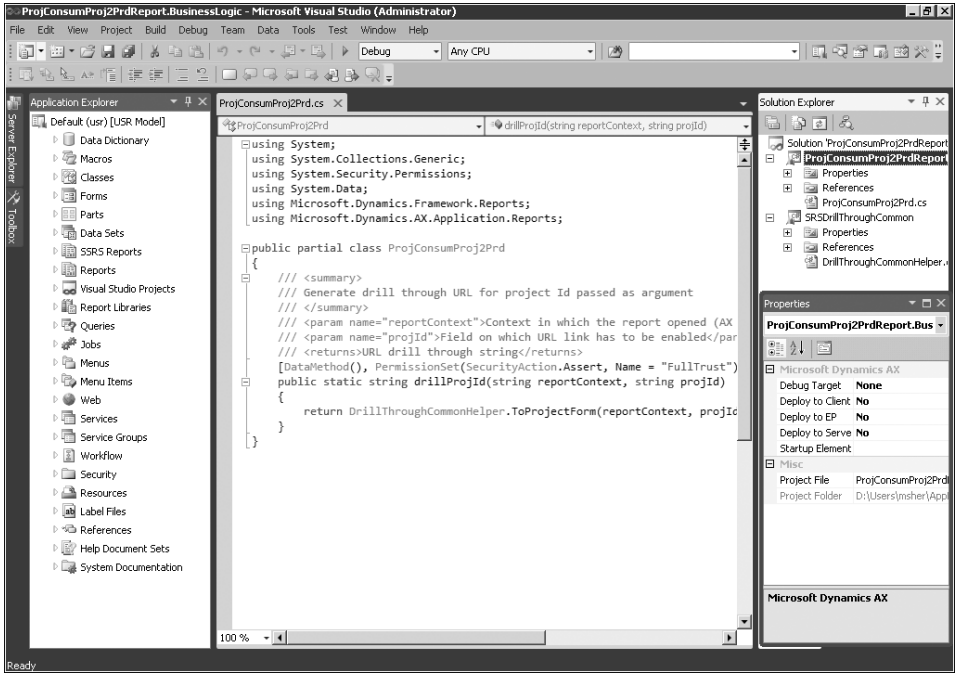

FIGURE 3-4 Application Explorer with a Microsoft Dynamics AX project open.

You can then create a C# representation of the table by dragging the table node from Application Explorer into the project.

After you drag the table node into the Visual Studio project, you will find an entry in the project that represents the table. The items that you drag into the project in this way are now available to code against in C#, just as though they had been written in C#. This happens because the drag operation creates a proxy for the table under the covers; this proxy takes care of the plumbing required to communicate with the Microsoft Dynamics AX system, while presenting a high-fidelity managed interface to the developer.

You can now proceed by putting the missing pieces into the C# code to write the data into the table. Modify the code as shown in the following example:

```
DataTable table = ds.Tables["Customers"];
var customers = new ReadFromExcel.CustomersFromExcel();
foreach (DataRow row in table.Rows)
{
    string name = row["Name"] as string;
    DateTime d = (DateTime)row["Date"];
    customers.Name = name;
    customers.Date = d;
    customers.Write();
}
```

Note The table from Microsoft Dynamics AX is represented just like any other type in C#. It supports IntelliSense, and the documentation comments that were added to methods in X++ are available to guide you as you edit.

The data will be inserted into the CustomersFromExcel table as it is read from the ADO.NET table that represents the contents of the spreadsheet. However, before either the client or the server can use this code, you must deploy it. You can do this by setting the properties in the Properties window for the Microsoft Dynamics AX project in Visual Studio. In this case, the code will run on the client, so you set the *Deploy to Client* property to *Yes*. There is a catch, though: you cannot deploy the assembly to the client when the client is running, so you must close any Microsoft Dynamics AX clients prior to deployment.

To deploy the code, right-click the Visual Studio project, and then click Deploy. If all goes well, a *Deploy Succeeded* message will appear in the status line.

Note You do not have to add a reference to the assembly because a reference is added implicitly to projects that you add to the AOT. You only need to add references to assemblies that are not the product of a project that has been added to the AOT.

As soon as you deploy the assembly, you can code against it in X++. The following example illustrates a simple snippet in an X++ job:

```
static void ReadCustomers(Args _args)
{
    ttsBegin;
    Contoso.ExcelReader::ReadDataFromExcel(@"c:\Test\customers.xlsx");
    ttsCommit;
}
```

When this job runs, it calls into the managed code and insert the records into the Microsoft Dynamics AX database.

Debug managed code

To ease the process of deploying after building, Visual Studio properties let you define what happens when you run the Microsoft Dynamics AX project. You manage this by using the *Debug Target* and *Startup Element* properties. You can enter the name of an element to execute—typically, a class with a suitable main method or a job. When you start the project in Visual Studio, it will create a new instance of the client and execute the class or job. The X++ code then calls back into the C# code where breakpoints are set. For more information, see "Debugging Managed Code in Microsoft Dynamics AX" at *http://msdn.microsoft.com/en-us/library/gg889265.aspx*.

An alternative to using this feature is to attach the Visual Studio debugger to the running Microsoft Dynamics AX client (by using the Attach To Process menu item on the Debug menu in Visual Studio). You can then set breakpoints and use all of the functionality of the debugger that you normally would. If you are running the Application Object Server (AOS) on your own computer, you can attach to that as well, but you must have administrator privileges to do so.

Important Do not debug in a production environment.

Proxies

As you can see, wiring up managed code to work with Microsoft Dynamics AX is quite simple because of the proxies that are generated behind the scenes to represent the Microsoft Dynamics AX tables, enumerations, and classes. In developer situations, it is standard to develop the artifacts in Microsoft Dynamics AX iteratively and then code against them in C#. This process is seamless because the proxies are regenerated by Visual Studio at build time, so that they are always synchronized with the corresponding artifacts in the AOT; in other words, the proxies never become out of date. In this way, proxies for Microsoft Dynamics AX artifacts differ from Visual Studio proxies for web services. These proxies are expected to have a stable application programming interface (API) so that the server hosting the web service is not contacted every time the project is built. Proxies are generated not only for the items that the user has chosen to drop onto the *Project* node as described previously. For instance, when a proxy is generated for a class, proxies will also be generated for all of its base classes, along with all artifacts that are part of the parameters for any methods, and so on.

To see what the proxies look like, place the cursor on a given proxy name in the code editor, such as *CustomersFromExcel* in the example, right-click, and then click Go To Definition (or use the convenient keyboard shortcut F12). All of the proxies are stored in the Obj/Debug folder for the project. If you look carefully, you will notice that the proxies use the Microsoft Dynamics AX Business Connector to do the work of interfacing with the Microsoft Dynamics AX system. The Business Connector has been completely rewritten from the previous version to support this scenario; in older versions of the product, the Business Connector invariably created a new session through which the interaction occurred. This is not the case for the new version of the Business Connector (at least when it is used as demonstrated here). That is why the transaction that was started in the job shown earlier is active when the records are inserted into the table. In fact, all aspects of the user's session are available to the managed code. This is the crucial difference between authoring business logic in managed code and consuming the business logic from managed code. When you author business logic, the managed code becomes an extension to the X++ code, which means that you can crisscross between Microsoft Dynamics AX and managed code in a consistent environment. When consuming business logic, you are better off using the services framework that Microsoft Dynamics AX provides, and then consuming the service from your application. This has big benefits in terms of scalability and deployment flexibility.

Figure 3-5 shows how the Business Connector relates to Microsoft Dynamics AX and .NET application code.

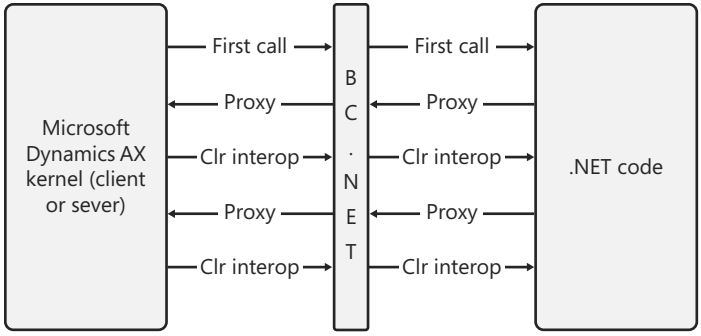

FIGURE 3-5 Interoperability between Microsoft Dynamics AX and .NET code through the Business Connector.

To demonstrate the new role of the Business Connector, the following example opens a form in the client that called the code:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace OpenFormInClient
{
    public class OpenFormClass
    {
        public void DoOpenForm(string formName)
        {
            Args a = new Args();
            a.name = formName;
            var fr = new FormRun(a);
            fr.run();
            fr.detach();
        }
    }
}
```

In the following example, a job is used to call managed code to open the CustTable form:

```
static void OpenFormFromDotNet(Args _args)
{
    OpenFormInClient.OpenFormClass opener;
    opener = new OpenFormInClient.OpenFormClass();
    opener.DoOpenForm("CustTable");
}
```

Note The *FormRun* class in this example is a kernel class. Because only an application class is represented in Application Explorer, you cannot add this proxy by dragging and dropping as described earlier. Instead, drop any class from Application Explorer onto the Visual Studio project, and then set the file name property of the class to *Class.*<kernelclassname>*.axproxy*. In this example, the name would be *Class.FormRun.axproxy*.

This would not have been possible with older versions of the Business Connector because they were basically faceless clients that could not display any user interface. Now, the Business Connector is actually part of the client (or server), and therefore, it can do anything they can. In Microsoft Dynamics AX 2012 R2, you can still use the Business Connector as a stand-alone client, but that is not recommended because that functionality is now better implemented by using services (see Chapter 12). The Business Connector that is included with Microsoft Dynamics AX is built with .NET Framework 3.5. That means that it is easier to build the business logic with this version of .NET; if you cannot do that for some reason, you must add markup to the App.config file to compensate. If you are using a program that is running .NET Framework 4.0 and you need to use the Business Connector through the proxies as described, you would typically add the following markup to the App.config file for your application:

```
<configuration>
  <startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>
</configuration>
```

Hot swap assemblies on the server

The previous section described how to express business logic in managed code. To simplify the scenario, code running on the client was used as an example. This section describes managed code running on the server.

You designate managed code to run on the server by setting the *Deploy to Server* property for the project to Yes, as shown in Figure 3-6.

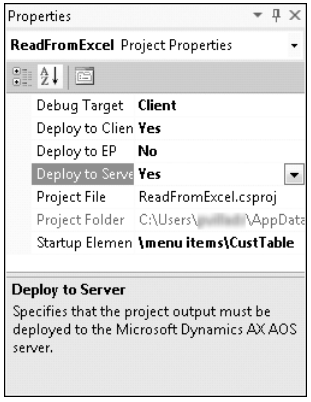

FIGURE 3-6 Property sheet showing the *Deploy to Server* property set to Yes.

When you set this property as shown in Figure 3-6, the assembly is deployed to the server directory. If the server has been running for a while, it will typically have loaded the assemblies into the current application domain. If Visual Studio were to deploy a new version of an existing assembly, the deployment would fail because the assembly would already be loaded into the current application domain.

To avoid this situation, the server has the option to start a new application domain in which it executes code from the new assembly. When a new client connects to the server, it will execute the updated code in a new application domain, while already connected clients continue to use the old version.

To use the hot-swapping feature, you must enable the option in the Microsoft Dynamics AX Server Configuration Utility by selecting the Allow Hot Swapping of Assemblies When The Server Is Running check box, as shown in Figure 3-7. To open the Microsoft Dynamics AX Server Configuration Utility, on the Start menu, point to Administrative Tools, and then click Microsoft Dynamics AX 2012 Server Configuration.

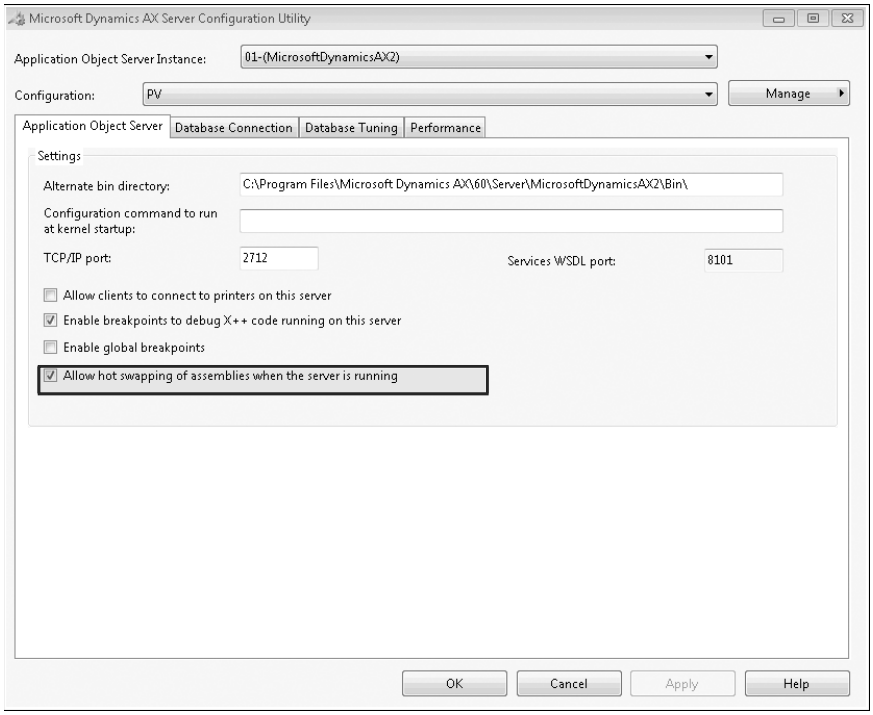

FIGURE 3-7 Allow hot swapping by using the Microsoft Dynamics AX Server Configuration Utility.

Note The example in the previous section illustrated how to run and debug managed code on the client, which is safe because the code runs only on a development computer. You can debug code that is running on the server (by starting Visual Studio as a privileged user and attaching to the server process as described in the "Debug managed code" section). However, you should never do this on a production server because any breakpoints that are encountered will stop all of the managed code from running, essentially blocking any users who are logged on to the server and processes that are running. Another reason you should not use hot swapping in a production scenario is that calling into another application domain extracts a performance overhead. The feature is intended only for development scenarios, where the performance of the application is irrelevant.

Index

Symbols and Numbers

.chm files, 545
.NET AJAX, 222
.NET Business Connector
  Enterprise Portal architecture, 8, 197–198
  Enterprise Portal security, 232–235
  Enterprise Portal, developing for, 231–232
  proxies, Enterprise Portal, 226–228
.NET CIL (common intermediate language), compile and run X++, 126–128
.NET CLR interoperability statement, X++ syntax, 96
.NET Framework. *See also* Windows Workflow Foundation (WF)
  assemblies, hot-swapping, 84–85
  author managed code, 77–84
  chart control mark-up elements, 291–292
  EP Chart Control tool, overview, 289
  legacy systems, overview, 73
  plug-ins, developing, 7
  processing architecture, 4–6, 8
  third-party assemblies, use of, 73–76
.NET Framework Common Language Runtime (CLR), 13
.rsds (data connection file), PowerPivot, 336
.xpo files, create a build, 71

A

abstract, method modifier, 119
acceptance test driven development (ATDD), 535–536
access control. *See also security*
  form permissions, 356–359
  privileges, creating, 359–361
  security framework overview, 353–356
  security roles, privileges and duties, 361–362
access operators, X++ expressions, 95
accounting framework
  extensions, 662
  MorphX model element prefixes, 663
  overview, 659–662
  process states, 662–663
  when to use, 662
Accounting Journalization Rule, 660
Accounting Policy, 659
accounting, element naming prefix, 22
acknowledgement message, workflows, 260
action menu items, workflow artifacts, 266
Action pane
  controls, creating, 174–176
  details form design, 151–152
  Enterprise Portal, navigation form design, 156–157
  Enterprise Portal, web parts, 199
  model-driven list pages, creating, 218
  transaction details form design, 155
Action pane, list page design, 149
ActionMenuItemClicked, 212–213
ActionMenuItemClicking, 212–213
actions, element actions in AOT, 25–26
Active Directory
  Integrated Windows Authentication, 352–353
  security role assignments, 355
ActiveMode, 204
ActiveSectionIndex, 204
Activity, operations resource framework, 650–653
add-ins, Microsoft Office, 189–190
add-on functionalities
  delegates, X++ syntax, 120–122
  naming of, 21–22
address bar, navigation layer forms, 141
address book, 637
addresses, element prefix, 22
addRuntimeTask, 624
addTask, batch jobs, 624
aggregate, X++ select statements, 102
Aggregation property, associations, 48–49

AIF (Application Integration Framework)
custom services, creating, 388–391
overview, 386
send framework, 411–414
services, publishing, 400–401
AIF Document Service Wizard
artifacts, creating, 393
creating document services, 395–397
opening, 29
AifCollectionTypeAttribute, 391
AifDocumentService, 393
AifEntityKeys, 394
AJAX, Enterprise Portal development, 222
AllowDelete, AxGridView, 207
AllowEdit, AxGridView, 207
AllowFormCompanyChange, window type, 174
AllowGroupCollapse, AxGridView, 207
AllowGrouping, AxGridView, 207
AllowSelections, AxGridView, 207
ALM, Visual Studio, 534, 540–543
alternate keys
date-effective framework, 602–603
overview, 587–588
AmountMST, 316–317, 328–332
Analysis Currency, 329–332
Analysis Services Projects, modifying prebuilt projects, 319–323
AnalysisDimensionLabel, 326–327
AnalysisIdentifier, 326–327
AnalysisKeyAttributeLabel, 326–327
AnalysisMeasureGroupLabel, 326–327
analytics. *See also* Business Intelligence (BI)
analytic content, configuring, 310–311
Business Overview and KPI List web parts, 341–345
Excel reports, 340
overview, 333–335
presentation tools, choice of, 335
Report Builder, 346
SQL Server Power View, 335–340
Visual Studio tools, 346–349
anytype, 90, 93
AOD (Application Object Data) files, 687–688
AOS (Application Object Server)
architecture, 7
batch jobs, debugging, 629–630
configuration, performance and, 463–464
Help system, 549
processing architecture, 3–5, 7
AOSAuthorization, coding table permissions, 369–371
aosValidateDelete, 436
aosValidateInsert, 438–439
aosValidateRead, 433–434, 436
aosValidateUpdate, 433–434
AOT (Application Object Tree) modeling tool
AxDataSource, 203–204
creating elements, 23
element actions in, 25–26
element layers and models, 26
Enterprise Portal, developing for, 217
Jobs, overview, 88
modifying elements, 23–25
navigation in, 21–23
overview, 20
PageTitle, 200
processing architecture, 6–7
publishing services, 400–401
refreshing elements in, 25
security artifacts, developing, 356–363
Table Browser, overview, 52–53
third-party DLLs, referencing, 75–76
Toolbar, Enterprise Portal web part, 200
APIs (application programming interfaces)
AIF Send, 411–414
Batch API, 623–625
code access security (CAS), 124–126, 371–372
document services, overview, 392
Enterprise Portal architecture, 196–198
model store, 703–704
reflection, overview, 669–670
table data, reflection API, 673–676
treenodes, 680–685
API exceptions, dangerous, 41
application development environment, 6–7
application domain frameworks
accounting framework
extensions, 662
MorphX model element prefixes, 663
overview, 659–662
process states, 662–663
when to use, 662
dimension framework
constrain combinations of values, 656
create values, 656–657
extensions, 657
overview, 654–656
physical table references, 659
query data, 658–659

operations resource framework
extensions, 652–653
MorphX model element prefixes, 654
overview of, 648–652
when to use, 652
organization model framework
custom operating units, creating, 639–640
hierarchy designer, extending, 642
integration with other frameworks
application modules, 637–638
organization hierarchies, 635–637
organization types, 634–635
overview, 634
scenarios, modeling, 638–639
overview, 633–634
product model framework
extension of, 647–648
overview, 643–647
when to use, 647
source document framework
extensions, 666–667
MorphX model element prefixes, 667
overview, 664–665
when to use, 665
application frameworks, element naming prefix, 23
Application Integration Framework
custom services, creating, 388–391
overview, 386
send framework, 411–414
services, publishing, 400–401
application integration services, architecture, 8
application meta-model architecture
application data element types, 10–11
code element types, 13
documentation and resource element types, 16
license and configuration element types, 16–17
MorphX user interface control element types, 11–12
overview, 9
role-based security element types, 14
services element types, 13
web client element types, 14–15
workflow element types, 12–13
application models
creating models, 693–694
element IDs, 692–693
layers, 688–690
model store API, 703–704
models, overview, 690–692
moving from test to production, 700–703
overview, 687–688
publishing, preparing for, 694–699
upgrading, 699–700
Application Object Data (AOD) files, 687–688
Application Object Server (AOS)
architecture, 7
batch jobs, debugging, 629–630
configuration, performance and, 463–464
Enterprise Portal architecture, 197–198
Help system, 549
processing architecture, 3, 5, 7
Application Object Tree (AOT) modeling tool
AxDataSource, 203–204
creating elements, 23
element actions in, 25–26
element layers and models, 26
Enterprise Portal, developing for, 217
Jobs, overview, 88
modifying elements, 23–25
navigation in, 21–23
overview, 20
PageTitle, 200
processing architecture, 6–7
publishing services, 400–401
refreshing elements in, 25
security artifacts, developing, 356–363
Table Browser, overview, 52–53
third-party DLLs, referencing, 75–76
Toolbar, Enterprise Portal web part, 200
application platform, architecture of, 4–9
application programming interfaces (APIs)
AIF Send, 411–414
Batch API, 623–625
code access security (CAS), 124–126, 371–372
document services, overview, 392
Enterprise Portal architecture, 196–198
model store, 703–704
reflection, overview, 669–670
table data, reflection API, 673–676
treenodes, 680–685
ApplicationHelpOnTheWeb, 548
applications, processing architecture, 4–5
approvals, workflow artifacts, 264
approvals, workflow elements, 252
architecture
application platform, 6–9
Business Intelligence (BI), 299–300
Enterprise Portal, 196–198

architecture *(continued)*
five-layer solution architecture, overview, 4–6
overview, 3–4
report execution sequence, 278–279
security framework, overview, 351–356
service-oriented architecture, 386
workflow architecture, 256–262
area pages, designing, 144–146
arithmetic operators, X++ expressions, 95
artifacts, workflows, 264–265
ASP.NET
AxDataSource, 203–204
chart control mark-up elements, 291–292
datasets, Enterprise Portal, 201–203
Enterprise Portal architecture, 196–198
Enterprise Portal, AJAX, 222
EP Chart Control Tool, overview, 289
error handling, 231–232
GridView, 207
processing architecture, 4–6
UpdatePanel, 215
User control web part, 201
validation, 231
ViewState, Enterprise Portal, 228–229
assemblies, hot-swapping, 84–85
assert, code access security, 126
Asset, element prefix, 22
assignment statements, X++ syntax, 96
associated forms, permissions, 358
associations, metadata, 163
asynchronous mode, SysOperations, 468
ATDD (acceptance test driven development), 535–536
attributes
SysTest framework, new features, 527–533
UML associations, 48–49
X++ syntax, 123–124
Attributes, product model framework, 646
authentication
Enterprise Portal architecture, 197
Enterprise Portal, security, 232–235
models, signing, 696–697
security framework overview, 351–356
Authenticode, 696–697
authorization, security framework overview, 351–356
Auto variables, overview, 187
autogeneration, buttons, 220
auto-inference, form permissions, 356–359
AutoLookup, coding, 188–189
automated decisions, workflow elements, 253
automated tasks, workflow artifacts, 265
automated tasks, workflow elements, 253
AutoQuery, 170–172
autorefresh, 25
AutoSearch, metadata property, 170
avg
sample select statement code, 102
X++ select statements, 102
Ax, element prefix, 22
AxActionPanel, 211–212
AxaptaObjectAdapter, 204
AxBaseValidator, 231
AxBaseWebPart, 224–225
AxColumn, 205
AxCommon, 393–394
AxContentPanel, 215
AxContext, 223–225
AxContextMenu, 209
Axd documents
business document updates, 407–409
document services artifacts, 392–393
overview, 392
Axd, element naming prefix, 22
AxDataSource, 201–204, 225
AxDataSourceView, 203, 209
AxDatePicker, 216
AxDateSource, 219–221
AxDateTimeValueFormatter, 230
AxDateValueFormatter, 230
AxdDocument class, 393–394
AxdSend API, 412–414
AxEnumValueFormatter, 230
AxExceptionCategory, 231–232
AxFatal, exception handling, 231
AxFilter, 209–210
AxForm, 204, 231
AxFormPart, 216
AxGridView, 207, 210, 215, 231
AxGroup, 205–206, 215
AxGuidValueFormatter, 230
AxHierarchicalGridView, 208
AxInfoPart, 216
AxInternalBase, 395
AxLabel, 229–230
AxLookup, 210–211
axmodel, 687. *See also* model store
axmodelstore, 688. *See also* model store
AxMultiColumn, 205

AxMultiSection, 204
AxNumberValueFormatter, 230
AxPartContentArea, 216
AxPopup controls, 213–215
AxPopupBaseControl, 213–215
AxPopupChildControl, 213–215
AxPopupField, 214–215
AxPopupParentControl, 213–215
AxRealValueFormatter, 230
AxReportViewer, 216
AxSection, 204
AxStringValueFormatter, 230
AxTable class, 393–395
AxTableContext, 223–225
AxTableDataKey, 224–225
AxTimeValueFormatter, 230
AxToolbar, 212–213
AxToolbarButton, 212
AxToolBarMenu, 212
AxUpdate Portal, 239–240
AXUtil
    element IDs, 692
    importing models, 697–699
    model store API, 703–704
    models, upgrading, 699–700
AxValueFormatter, 230
AxValueFormatterFactory, 230
AxValueFormatValidator, 231
AxViewContext, 223–225
AxViewDataKey, 224–225

B

backing entity type, adding, 657
base enum elements, defined, 10
base enumeration types, 88, 93
Batch API, 623–625
Batch class, 49–50
batch framework
    Batch API, using, 623–625
    batch group, creating, 626–627
    batch jobs, creating, 618–625
    batch server, configuring, 625–626
    batch-executable class, creating, 616–617
    common uses of, 614–615
    debugging batch jobs, 629–631
    managing batch jobs, 628–629
    overview, 613–614
    performance and, 466–467, 615
Batch Job Form
    creating batch jobs, 619–622
    overview, 613
Batch Tasks form, 621–622
BatchHeader, 623–625
BatchRunable interface, 495–496
best practices
    exception handling, 105
    variable declaration syntax, 94
    X++ syntax, 93
Best Practices tool
    custom rules, 42–43
    errors and warnings, suppressing, 41–42
    overview, 20, 39–40
    rules, 40–41
    Trustworthy Computing, 372–373
    XML documentation, 116
BI. *See* Business Intelligence (BI)
bill of materials, element prefix, 22
binding
    AxToolbar, 212
    BoundField, 215
    chart controls, binding data series, 292–294
    chart controls, binding to dataset, 292
    control data binding, 173
    datasets, Enterprise Portal, 201
    field-bound controls, 178–179
    method-bound controls, 179
    object type, 92
bitwise operators, X++ expressions, 95
BOM, element prefix, 22
boolean
    value types, overview, 88
    variable declaration syntax, 93
BoundField, overview, 215
boxing, 110
break statement, X++ syntax, 96
Break, exception handling, 106
breakpoint
    AOS configuration, performance and, 463–464
    breakpoint statement, X++ syntax, 96
    shortcut keys, X++ code editor, 32
browsers, Enterprise Portal architecture, 197–198
buf2con, 425–426
built-in collection types, 88
built-in primitive type, 88
Business Connector, authoring managed code, 77–84
business documents, element prefix, 22

Business Intelligence (BI)
analytic content, configuring, 310–311
components of, 299–300
cubes, creating
generate and deploy cubes, 328–333
KPIs and calculations, adding, 333
metadata, defining, 325–328
requirements, identifying, 324–325
cubes, customizing, 311–319
cubes, extending, 319–323
customizing, overview, 309–310
displaying content in Role Centers
Business Overview and KPI List web parts, 341–345
Excel reports, 340
overview, 333–335
presentation tools, choice of, 335
Report Builder, 346
SQL Server Power View reports, 335–340
Visual Studio tools, 346–349
Enterprise Portal, web parts, 199
overview, 299
prebuilt BI solution, implementing, 301–309
properties, 326–327
Business Intelligence Development Studio, 323
business logic
overview, 188
workflows, creating, 264–265
Business Overview
Enterprise Portal, web parts, 199
Role Center displays, 341–345
business processes, defined, 245–246. *See also* workflow
business unit, defined, 635
buttons
action controls, creating, 174–176
action controls, overview, 175
AxToolbarButton, 212
details page, autogeneration, 220
Enterprise Portal, AJAX, 222
model-driven list pages, creating, 218–219
bytecode, overview, 87

C

C#, authoring managed code, 77–84
caching
cacheAddMethod, 418–419
CacheDataMethod, 419
CacheLookup, 421, 446–452
declarative display method caching, 419
elements, refreshing, 25
Enterprise Portal, developing for, 223
EntireTable cache, 453–454
indexing and, 421
labels, 230
MetadataCache, 225–226
number sequence caching, 465
performance
overview, 446
record caching, 446–452
unique index join cache, 452
RecordViewCache, 454–455
Server Configuration form, 463
SysGlobalCache, 456
SysGlobalObjectCache, 456
update conflicts, 409
CAL (client access license), 376–383
calculated measures, adding to cubes, 333
calendars, customizing cubes, 314–315
call stack, 88, 125–126
CallContext, consuming system services, 404–407
Called From, 420
camel casing, 93
cancel selection, shortcut key, 32
candidate key, alternate keys, 587–588
canGoBatchJournal, 617
canSubmitToWorkflow, 274
canSubmitToWorkflow, workflow artifacts, 266
Capability, 649–650
Caption property, forms, 160, 174
card sort, 145–146
CAS (code access security), 124–126, 371–372
casting statements, X++ syntax, 97
categories, product model framework, 646
changeCompany statement, X++ syntax, 98
ChangeGroupMode, 167
chart development tools, overview, 289
charts. *See* reporting
CIL (common intermediate language)
executing X++ as CIL, 466
troubleshooting, tracing, 487–488
claims-based authentication, 352–353
class element type, 13
class type, reference types, 89
Class Wizard, opening, 29
classDeclaration, attributes, 123–124

classes
attributes, X++ syntax, 123–124
fields, X++ syntax, 118
UML object models, 49–50
X++ syntax, overview, 117–118
classIdGet system function, reflection, 669, 672–673
ClearFieldValues, 214–215
client access license (CAL), 376–383
client access log, 490
client callbacks, eliminating, 424–425
client configuration, performance and, 464–465
client, method modifier, 119
Close button, autogeneration, 220
ClosePopup, 214–215
CLR (Common Language Runtime)
reference element types, 13
type conversions, 111
X++ interoperability, 108–112
CLRError, exception handling, 106
ClrObject, calling managed code, 76
code access security (CAS), 124–126, 371–372
code element types, overview, 13
code permission element type, 14
CodeAccessPermission, 125, 371–372
CodeAccessPermission.copy, 372
CodeAccessPermission.demand, 372
CodeAccessPermission.isSubsetOf, 372
CodeAccessSecurity, exception handling, 106, 372
coding. *See also* X++ programming language (code)
Auto variables, 187
business logic, 188
custom lookups, 188–189
form customization, overview, 184
method overrides, 184–186
collections list page example, 148–149
columns
AxColumn, 205
AxMultiColumn, 205
computed columns, creating views, 333
CombineXPOs, 71
CommandButton, 175
comments
inserting, shortcut key for, 32
X++ syntax, 115
common intermediate language (CIL)
executing X++ as CIL, 466
troubleshooting, tracing, 487–488
Common Language Runtime (CLR)
reference element types, 13
type conversions, 111
X++ interoperability, 108–112
common type, 90–91
Common, Area Page design, 145
Compare tool, 20, 54–59
compilation
Compiler output window, 38
compiling and running X++ as .NET CIL, 126–128
EB Web Applications, 220
errors, shortcut keys, 32
errors, use of semicolon, 95
intrinsic functions, reflection, 670–671
overview, 37–39
shortcut keys, X++ code editor, 32
Compiler tool, 20
compound statement, X++ syntax, 97
computed columns, views, 333
con2buf, 425–426
ConceptNum, 671
ConceptStr, 671
Concrete Source Documents, 664–665
conditional operators, X++ expressions, 95
conditions, workflow document classes, 268–270
configuration hierarchy, license codes, 378
configuration key element type, 16–17
configuration keys
Business Intelligence (BI), 309
license codes, 378–380
table inheritance hierarchy, 595
ConfigurationKey, temporary tables, 584
conflict resolution, project upgrades and, 29
Connect, Enterprise Portal web parts, 199
constrained table
data security policies, developing, 365–369
defined, 365
constructor encapsulation pattern, 129–130
constructor, defined, 120
consumer, eventing, 520
containers
converting table buffers, 425–426
variable declaration syntax, 93
content pane, navigation form design, 156–157
content pane, navigation layer forms, 142
ContentPage, window type, 174
context menu, AxContextMenu, 209
Context, Enterprise Portal development, 223–225
ContextMenuName, AxGridView, 207
context-sensitive Help topics, 561–562

ContextString, data security policies, 367
continue statement, X++ syntax, 97
controls
adding, overview, 172–173
control data binding, 173
Design node properties, 173–174
dialog box controls, 494
form permissions, 359
input controls, overview, 178–179
layout controls, overview, 176–178
ManagedHost control, 179–181
report elements, 282–285
runtime modifications, 174
conversationId, 412
CopyCallerQuery property, form queries, 172
Correction, date-effective tables, 605–606
CorrectPermissions, 360
COS, element prefix, 22
cost accounting, element prefix, 22
cost center, defined, 635
count, select statements, 102
Create New Document Service Wizard, 393
create, read, update, and delete (CRUD) permissions
forms, 356–359
menu items, 360
CreateNavigationPropertyMethods, 591–593
CreateNewTimePeriod, 605–606
CreatePermissions, 360
createRecord, 167
CRM, element prefix, 23
crossCompany, 99
CrossCompanyAutoQuery, metadata property, 169
Cross-reference tool
overview, 60–61
Cross-reference tool, overview, 20
CRUD (create, read, update, and delete) permissions
forms, 356–359
menu items, 360
cubes
calendars, customizing, 314–315
creating
generate and deploy cubes, 328–333
metadata, defining, 325–328
requirements, identifying, 324–325
currency conversion, 316–317
customizing, 311–319
extending, data source, 321
extending, DSV, 321
extending, external data sources, 322–323
extending, KPIs and calculations, 321
extending, measures and dimensions, 321
extending, overview, 319–321
financial dimensions, 313–314
languages, selecting, 315–316
SSAS deployment, 303–309
cues
cue element type, MorphX, 12
cue group element type, MorphX, 12
Cue Groups, creating, 218
CueGroupPartControl, 216
Enterprise Portal web part, 199
parts overview, 181
Role Center page design, 143
culture, Enterprise Portal formatting, 230
currency conversion
cubes, adding logic, 328–332
overview, 316–317
surrogate keys, 586–587
CurrentContextChanged, 224–225
CurrentContextProviderView, 224–225
CurrentDataAccess, 363
CurrentDate, time period filters, 345
Cust, element naming prefix, 22
custom rules, adding, 42–43
custom workflow providers, workflow artifacts, 266
customer contact, list page sample design, 146–148
Customer Group lookup, 210–211
customer relationship management, element prefix, 23
customers, element prefix, 22
customization
Business Intelligence (BI), 309–319
chart reporting, default override, 296
custom services, 388–391
delegates, X++ syntax, 120–122
deploying, 700
document services, overview, 392, 397–399
Enterprise Portal architecture, 196–198
model store API, 703–704
using code, overview, 184
web parts, 199–201

D

dangerous API exceptions, 41
data. *See also* metadata; also transaction performance
access to, Enterprise Portal, 225–226
Axd queries, creating, 395–396

chart controls, binding to, 292–294
consistency, date-effective framework, 604–606
control data binding, 173
cubes, metadata selection, 325–328
data contracts, custom services, 389–390
data contracts, X++ collections as, 391
extensible data security policies, creating, 364–369
external data source integration, 322–323
form data sources, 164–169
form method overrides, 185–186
metadata, form data source properties, 168–169
Microsoft Office add-ins, 189–190
report elements, design of, 282–285
security framework overview, 351–356
valid time state tables, use of, 362–363
data binding
BoundField, 215
datasets, Enterprise Portal, 201
Data DictionaryPerspectives, 325–326
data model, Cross-reference tool, 60
data object, document services, 393
data processing extensions, reports, 288
data source, element prefix, 22
data tier
architecture, 7
Business Intelligence (BI), 299–300
dataAreaId, indexing tips, 475–476
data-aware statements, X++ syntax, 99–104
database statements, X++ syntax, 99–104
database tier. *See also* transaction performance
alternate keys, 587–588
date-effective framework, 601–606
full-text support, 606–607
overview, 577
QueryFilter API, 607–612
surrogate keys, 585–587
table inheritance, 594–599
table relations, 588–593
temporary tables
creating, 583–585
InMemory tables, 578–582
TembDB temporary tables, 582–583
Unit of Work, 599–601
DataBound, AxGridView, 207
DataContractAttribute, 390, 495
DataKeyNames, AxForm, 204
DataMember, 203–204, 207
DataMemberAttribute, 390, 495
DataSet, details page, 219–221
datasets
AxGridView, 207
Enterprise Portal, 201–203
DataSetView, 204, 209, 225
DataSetViewRow, 204
DataSourceControl, 203–204
DataSourceID
AxDataSource, 203–204
AxForm, 204
AxGridView, 207
DataSourceName, Auto variables, 187
DataSourceView, 203
date-effective framework
data consistency, run-time support, 604–606
data retrieval support, 603–604
overview, 601
relational modeling, 601–603
dates
AxDatePicker, 216
calendars, customizing, 314–315
currency conversion logic, 330–332
DATE, 314
Date Dimensions, 314–315
date effectivity, overview, 168
date, value types, 88
date, variable declaration syntax, 94
valid time state tables, 355
DDEerror, exception handling, 106
Deadlock, 106
debugging
batch jobs, 629–631
Debug Target, 81–82
Debugger tool, MorphX, 20, 43–47
extensible data security policies, 368–369
managed code, 81–82
security, 373–375
Table Browser tool, 52–53
declarations, Help topics, 552–553
declarative display method caching, 419
decoupling, events, 521
Default Account, defined, 655–656
Default Dimensions, defined, 655
defaulting logic, document services, 399
delegate, eventing, 521–524
delegate, method modifier, 119
delegates, X++ syntax, 120–122
delete
current selection, shortcut keys, 32
InMemory temporary tables, 582
table relations, 588–593

delete permissions
forms, 356–359
menu items, 360
delete, deleting, and deleted methods, 168
delete_from, 104, 428, 435–436
DeletePermissions, 360
department, defined, 635
dependent workflow artifacts, 264–265
deployment. *See also* Microsoft Dynamics AX services
assemblies, hot-swapping, 84–85
cubes, 328–333
models, 700–703
third-party DLLs, 75–76
Version Control, 63–64
web part pages, 239–240
Derived Data Sources, 165–167
derived table, creating, 594
design definition, report elements, 282–285
Design node, control properties, 173–174
design patterns, X++, 128–133
details form
designing, 150–153
transaction details form, designing, 153–155
details page, creating, 219–221
DetailsFormMaster template, 161
DetailsFormTransaction template, 161
DeveloperDocumentation, 548
development environment. *See* application meta-model architecture; MorphX
development tools, element prefix, 23
device CAL, 376–383
dialog box controls, 494
Dialog template, 161
dictionary, reflection API, 670, 676–680
digital signature, assembly names, 74–75
Dimension Attribute
dimension framework overview, 654–656
query data, 658–659
Dimension Attribute Sets, defined, 655
Dimension Derivation Rule, 660
dimension framework
constrain combinations of values, 656
create values, 656–657
extensions, 657
overview of, 654–656
physical table references, 659
query data, 658–659
Dimension Sets, defined, 655
DimensionConstraintNode, 656
DimensionDefaultingService, 656–657
dimensions, cubes, 321
DimensionStorage, 656–657
DimensionValidation
validateByTree, 656
Dir, element prefix, 22
directory, element prefix, 22
DirPartyTable_FK, 592
DirPersonName, 601–603
disableCache, 450
display menu items, workflow artifacts, 266
display, method modifier, 119
DisplayGroupFieldName, AxGridView, 207
DLLs (dynamic-link libraries)
referencing managed DLLs, 75–76
using third-party assemblies, 73–76
do while statement, X++ syntax, 97
document class, workflows, 268–270
document files, Help server, 547–549
document head, Help content, 553–556
document services
artifacts, 392–393
AxdDocument class, 393–394
AxTable classes, 395
creating, 395–397
customizing, 397–399
overview, 392
document set, Help system, 571
documentation element type, 16
documents
element naming conventions, 22
processing architecture, 5
doDelete, 428
doInsert, 428
doUpdate, 428
DropDialog template, 161
DropDialogButton, 175
DSV, overview, 321
DuplicateKeyException, 107
DuplicateKeyException-NotRecovered, 107
duties, security framework overview, 353–356
duty element type, overview, 14
dynalinked data sources
LinkType, metadata property, 169
Microsoft Dynamics AX client, 164–169
Dynamic Account, defined, 656
dynamic role assignment, 355
dynamic-link libraries (DLLs)
referencing managed DLLs, 75–76
using third-party assemblies, 73–76

DynamicsAxEnterprisePortal, 237
DynamicsAxEnterprisePortalMOSS, 237
DynamicsAxWebParts, 237
DynamicsLeftNavProvider, 235
DynamicsMOSSTopNavProvider, 235
DynamicsSearch, 237
DynamicsTopNavProvider, 235

E

economic resources, element prefix, 22
EcoRes, element prefix, 22, 647–648
editing
    AllowEdit, AxGridView, 207
    details form design, 152
    edit permissions, menu items, 360
    edit, method modifier, 119
EDT. *See* extended data type (EDT)
EDT Relation Migration tool, 589
EffectiveAccess, form permissions, 358
EffectiveBased, date-effective tables, 605–606
element IDs
    application models, 692–693
    model deployment, 702–703
element, Auto variables, 187
elements
    actions in AOT, 25–26
    Compare tool, 54–59
    creating in AOT, 23
    Cross-reference tool, overview, 60–61
    history of, viewing, 69
    layers and models in AOT, 26
    modifying in AOT, 23–25
    naming conventions, 21–22
    pending, Version control, 70
    Projects, creating, 27
    property sheet, overview, 30–31
    refreshing elements, AOT, 25
    upgrades, conflict resolution, 29
    version control system life cycle, 64–65
    Version Control tool, overview, 62–64
    workflow elements, 252–253
Enable block selection, 32
encryption, Enterprise Portal, 235
endpoints, associations, 49
Enterprise Portal. *See also* Enterprise Portal, developing for
    AOT elements, 201
    architecture, 8, 196–198
    authentication, overview, 352–353
    AxActionPanel, 211–212
    AxColumn, 205
    AxContentPanel, 215
    AxContextMenu, 209
    AxDataSource, 203–204
    AxDatePicker, 216
    AxFilter, 209–210
    AxForm, 204
    AxFormPart, 216
    AxGridView, 207
    AxGroup, 205–206
    AxHierarchicalGridView, 208
    AxInfoPart, 216
    AxLookup, 210–211
    AxMultiColumn, 205
    AxMultiSection, 204
    AxPartContentArea, 216
    AxPopup controls, 213–215
    AxReportViewer, 216
    AxSection, 204
    AxToolbar, 212–213
    BoundField, 215
    charting controls
        chart development tools, 289
        data series, binding, 292–294
        EP Chart Control, creating, 290–291
        mark-up elements, 291–292
        overview, 289
    CueGroupPartControl, 216
    datasets, 201–203
    overview, 195
    presentation tier architecture, 8
    processing architecture, 5
    Visual Studio development environment, 7
    web client experience, designing, 155–157
    web parts, overview of, 199–201
    workflow menu items, 252
Enterprise Portal, developing for
    AJAX, 222
    Context, 223–225
    data, access, 225
    details page, creating, 219–221
    Enterprise Search, 240–243
    error handling, 231–232
    formatting, 230
    labels, 229–230
    metadata, access to, 225–226
    model-driven list page, creating, 217–219
    overview, 216–217

Enterprise Portal, developing for, *(continued)*
proxy classes, 226–228
security, 232–235
session disposal and caching, 223
SharePoint integration
Enterprise Search, 240–243
site definitions, page templates, and web parts, 237–239
site navigation, 235–236
themes, 243
web part page, import and deploy, 239–240
validation, 231
ViewState, 228–229
Enterprise Search
Enterprise Portal and SharePoint integration, 240–243
EntireTable caching, 421, 453–454, 463
entity relationship data model, 51
enumeration types, 88
EPApplicationProxies, 226–228
EP Chart Control
creating, 290–291
data series binding, 292–294
mark-up elements, 291–292
overview, 289
EPSetupParams, 237
EP User Control with Form template, 219–221
EP Web Application Project template, 219–221
Error, exception handling, 107
errors
compiler errors, 37–39
error handling, Enterprise Portal development, 231–232
error warnings, Enterprise Portal web parts, 200
suppressing, Best Practices tool, 41–42
Esc key, 32
evalBuf, code access security, 124–126
event handlers
InMemory temporary tables, 582
pre- and post-, X++ syntax, 122–123
pre- and post-events, overview, 522–523
workflow artifacts, 266
workflows, 252
workflows, state management, 266–267
event logging, processing architecture, 5
event payload, defined, 521
eventhandler, delegate subscription, 121–122
eventing extensibility pattern
delegates, 521–522
event handlers, 523–524
eventing example, 524–525
overview, 520–521
pre and post events, 522–523
Excel
architecture, 9
authoring managed code example, 77–84
PowerPivot, SQL Server Power View, 335–340
reports, displaying in Role Centers, 340
templates, building, 190–193
Excel Services for SharePoint, report display, 340
Excel Services web part, report display, 340
exception handling, X++ syntax, 105–108
exceptions, dangerous APIs, 41
exchange rates, 329–332
Exchange Rates By Day, 331–332
execute current element, 32
execute editor script, 32
ExecutePermission, 371–372
executeQuery method, 172
ExecuteQuery, AxFilter, 209
ExistJoin, 164
exists, 103
exists method, 131–132
ExpansionColumnIndexesHidden, 207
ExpansionTooltip, AxGridView, 207
expense reports, 638
ExplicitCreate, 168
exporting elements, Compare tool, 56
exporting models, 695–696
expressions
expression builder, workflow document, 250, 269
X++ syntax, 95
extended data type (EDT)
AmountMST, currency conversions, 316–317
AxLookup, 210–211
defined, 10
overview, 88, 92–93
table relations, 588–590
value types, overview, 88
variable declaration syntax, 94
extensible data security framework (XDS)
debugging data security policies, 368–369
organization model framework integration, 638
policies, creating, 364–369
temporary table constructs, 368
extension framework
creating extensions, 517–518
extension example, 518–520
extensions, SQL Server Reporting Services (SSRS), 8

external name, service contracts, 389–390
ExternalContextChanged, 224–225
ExternalContextProviderView, 224–225

F

F1, context-sensitive Help, 561–562
FactBox
    AxPartContentArea, 216
    details form design, 152
    Enterprise Portal, navigation form design, 156–157
    form parts, overview, 182
    list page design, 150
    model-driven list pages, creating, 218
    MorphX user interface control element type, 12
    navigation layer forms, 142
    transaction details form design, 155
FastTabs, details form design, 150–153
FastTabs, TabPage controls, 177–178
field level properties, 327–328
field lists, performance and, 456–462
field-bound controls, 178–179. *See also* controls
fields, Help topics, 559–561
fields, X++ syntax, 118
file transfers. *See also* .NET Framework
    legacy systems, overview, 73
FileIOPermission, 371–372
filters
    AxDataSource, 203–204
    AxFilter, 209–210
    Projects, automatic generation of, 28
    QueryFilter, 172, 607–612
    ResetFilter, 209
    ShowFilter, AxGridView, 208
    SystemFilter, 209
    SysTest framework, new features, 530–533
    time period, Business Overview web part, 344–345
    UserFilter, 209
final, method modifier, 119
financial dimensions, cubes, 313–314
financial dimensions, organization model framework, 637
find method, 131–132
Find tool, 20
    overview, 53–54
finished goods management. *See* product model framework
FireCurrentContextChanged, 224–225
FireExternalContextChanged, 224–225
firstFast, 99, 476
firstOnly, 100
firstOnly10, 100
firstOnly100, 100
firstOnly1000, 100
FISCALPERIODDATEDIMENSION, 314
flexible authentication, overview, 352–353
flush statement, X++ syntax, 97
folders
    Help content, publishing, 567–571
    Projects, automatic generation of, 27–29
for statement, X++ syntax, 97
forceLiterals, 100
forceNestedLoop, 100
forcePlaceholders, 100
forceSelectOrder, 100
foreign key relations, 590–593
foreign keys, surrogate, 168
form element type, MorphX, 12
form part element type, MorphX, 12
Form Parts, model-driven list pages, 218
form parts, overview, 182
form rendering, MorphX user interface controls, 11–12
Form.DataSources, 162
Form.Designs.Design, 162
Form.Parts, 162
formatting, Enterprise Portal, 230
FormDataSource.AutoQuery, 170
FormDataSource.cacheAddMethod, 418–419
FormDataSource.create method, 166
FormDataSource.init, 172
FormDataSource.query, 172
FormDataSource.queryRun.query, 172
forms. *See also* details form; also Enterprise Portal, developing for; also Purchase Order form; also Sales Order form; also user experience, designing
    debugging security, 373–375
    display and edit methods, cacheAddMethod, 418–419
    form metadata, 162–164
    form patterns, 160–162
    form queries, overview, 170–172
    method overrides, 184–186
    navigation layer forms, design basics, 141–142
    navigation layer forms, Enterprise Portal design, 156–157

forms *(continued)*
permissions, setting, 356–359
working with, overview, 159–160
FormTemplate, 218
forUpdate, 100, 446–452
Found, record caching, 446–452
FoundAndEmpty, record caching, 446–452
foundation application domain partition, 4–6
FutureDataAccess, 363

G

General Journal, 660
general ledger, element prefix, 23
generateOnly, 100, 368–369
generic group reference, 132–133
Generic Test Case, 541
GetClosePopupEventReference, 214–215
GetCurrent, 204
getDataContractInfo, 494
GetDataSet, 204
GetFieldValue, AxPopup, 214–215
GetOpenPopupEventReference, 213–215
getQueryName, 269
getServiceInfo, 494
getter methods, 592
global address book, element prefix, 22
Glossary, Help system, 548
Go to implementation, 32
Go to next method, 32
Grid control, overview of, 176, 178
GridColumnIndexesHidden, 207
GridView
AxGridView and, 207
BoundField, 215
group by, select statement code sample, 102
Group control, overview of, 176
GroupField, 207
GroupFieldDisplayName, 207
grouping, AxGridView, 207
GroupMask, 28–29
Groups, automatic project generation, 27–29
Guest accounts, Enterprise Portal, 232–235
GUID, AxGuidValueFormatter, 230

H

header view, transaction details form, 154
headers, XML insert shortcut key, 32
help documentation set element type, 16
Help system
creating content
add labels, fields, and menu items, 559–561
context-sensitive topics, 561–562
non-HTML topics, 565–567
overview, 550–552
table of contents, creating, 563–565
topics, create in HTML, 552–559
updating content, 562–563
Help server, 546–549
help text, label file, 229–230
help text, SysOperationHelpTextAttribute, 495
Help viewer, 546–547
Microsoft Dynamics AX client, 547
overview, 545–549
processing architecture, 8
publisher, 550
publishing content, 567–571
shortcut keys, X++ code editor, 32
summary page, 550–551
table of contents, 550
topics, 549–550
troubleshooting, 572–573
HiddenField, AxPopupField, 214–215
hierarchy
organizational model framework, 635–637
type hierarchies, 89–93
hierarchy designer, extending, 642
HierarchyIDFieldName, 208
HierarchyParentIdFieldName, 208
history, batch jobs, 629
horizontal application domain partition, 4–6
hot-swapping assemblies, 84–85, 463–464
HRM/HCM, element prefix, 22
human resources management processes
element prefix, 22
organization model framework integration, 638
human workflows, defined, 246–249. *See also* workflow
HyperLinkMenuItem, model-driven list pages, 218

I

ICustomFormatter, 230
identifierStr, 671
if statement, X++ syntax, 97
IFormatProvider, 230
IIS (Internet Information Services), 8, 197–198

implementation patterns, X++, 128–133
ImplicitCreate, 168
ImplicitInnerOuter, 167
importing elements
    Compare tool, 56
    web part pages, 239–240
importing models, 697–699
Included Columns, indexing tips, 475–476
incremental search, shortcut key, 32
index
    ActiveSectionIndex, 204
    alternate keys, 587–588
    caching and, 421
    database query sample, 101
    Enterprise Search, 240–243
    ExpansionColumnIndexesHidden, 207
    GridColumnIndexesHidden, 207
    inMemory temporary tables, 423
    performance tips, 475–476
    record caching, 448–449
    SelectedIndexChanged, 208
    unique index join cache, 452
indexed sequential access method (ISAM), 577–582
IndexTabs, TabPage controls, 176–178
industry application domain partition, 4–6
info part element type, MorphX, 12
Info Parts, model-driven list pages, 218
info parts, overview, 181
Infolog
    Enterprise Portal, web parts, 200
    validation, 231
information dissemination, events, 521
information messages, Enterprise Portal web parts, 200
infrastructure callback, workflow, 261
inheritance
    RunBase, 510–511
    tables, 91, 165–167, 594–599
inheritance, metadata, 163–164
init method, Enterprise Portal datasets, 202
initializing, SysListPageInteractionBase, 218
Inline, WindowMode, 220
InMemory temporary tables, 422–423, 428, 578–585
InnerJoin, 164
input controls, overview, 178–179. *See also* controls
Inquiries, Area Page design, 145
insert method, transaction statement, 104
insert_recordset
    RowCount, 104
    sample statement, 104
    table hierarchies and, 428
    transaction performance and, 429–432
    transferring code into set-based operations, 442–444
insertDatabase, transaction performance, 437–439
InsertIfEmpty, metadata property, 169
installation, third-party DLLs, 75–76
InstanceRelationType, 594
int
    value types, overview, 88
    variable declaration syntax, 94
int64
    value types, overview, 88
    variable declaration syntax, 94
Integrated Windows Authentication, 352–353
integration tier, Business Intelligence, 299–300
IntelliMorph, 8, 33
IntelliSense, 73–76
InteractionClass, 218–219
interactive functions, reports, 294–295
interfaces
    code access security (CAS), 124–126
    fields, syntax, 118
    label file, 229–230
    methods, syntax, 118–120
    reference types, overview, 89
    UML object models, 49–50
    X++ syntax, overview, 117–118
inter-form dynalinks, 165
Internal, exception handling, 107
International Financial Reporting Standards, segregation of duties, 354
Internet Information Services (IIS), 8, 197–198
InteropPermission, 371–372
intra-form dynalinks, 165
intrinsic function, reflection, 669–671
Invent, element prefix, 22
inventory management, 22, 465. *See also* product model framework
Inventory Model Group, 645
is operator, 92
ISAM (indexed sequential access method), 577–582
IsConsumingMemory, 684
IsGetNodeInLayerSupported, 684
IsLayerAware, 684
IsLookUp, 327

IsModelElement, 684
IsRootElement, 684
isTmp, 582
IsUtilElement, 684
IsVCSControllableElement, 684
Item Group, 645

J

JMG, element prefix, 23
job element types, 13
Jobs, overview of, 88
joined data sources, Microsoft Dynamics AX client, 164–169
joins
    LinkType, metadata property, 169
    maximum number, 463
    select statement code sample, 102
JoinSource, metadata property, 169
Journalization Processor, 660–662
Journalization Rules, 660–662
Journals, Area Page design, 145

K

keywords, data-aware statement options, 99
KM, element prefix, 23
knowledge management, element prefix, 23
KPI
    adding to KPI List web part, 342–344
    cubes, creating, 333
    SSAS projects, 321
KPI List web part, Role Center displays, 341–345

L

Label editor tool
    creating labels, 35–36
    overview, 20, 33–35
    referencing X++ labels, 36–37
    shortcut keys, X++ code editor, 32
labels
    Enterprise Portal, developing for, 229–230
    Help topics, 559–561
    label file element type, 16
    Label Files, 33–34
    Label Files Wizard, 35
language
    cubes, customizing, 315–316
    formatting, Enterprise Portal, 230
    label editor, 33
    label editor, creating labels, 36
layer comparison, Project development tools, 29
layers, metadata
    models, overview, 690–692
    overview, 688–690
layout controls
    input controls, overview, 178–179
    overview of, 176–178
layout, hiding report columns, 286–288
Ledger Dimensions
    creating, 656–657
    defined, 655
Ledger, element prefix, 23
Left navigation, Enterprise Portal web part, 200
legacy systems. *See also* .NET Framework
    legacy data source integration, 323
    working with, 73
LegacyID, 692
legal entity, defined, 634
license code element type, 16–17
license keys, generating, 703–704
licensing
    CALs, types of, 381–382
    customization and, 383
    overview, 376–383
line view, transaction details form, 154
line-item workflows, overview, 253
links
    data sources, Microsoft Dynamics AX client, 164–169
    Help content, 558
    hiding, ShowLink, 236
    HyperLinkMenuItem, 218
LinkType, 164, 169
list definition element type, 15
list pages
    designing, 146–150
    model-driven list page, creating, 217–219
    performance optimization, 476
ListPage template, 161
ListPage, window type, 174
ListPageInteraction, 218
literalStr, 36–37, 671
loadData, 653

local function statement, X++ syntax, 98
localization
    label editor, 33
    label editor, creating labels, 36
logging, InMemory temporary tables, 582
logical operators
    logical approval and task workflows, 260–262
    X++ expressions, 95
LogonAs, Enterprise Portal security, 232–235
LogonAsGuest, Enterprise Portal security, 232–235
lookups
    AxLookup, 210–211
    custom, coding, 188–189
    table relations, 588–593
LookupType, 211
loop iterations, reducing, 477–478

M

macro element types, overview, 13
macros, X++ syntax, 113–115
Main Account Derivation Rule, 660
Main Account Dimension Attribute, 658–659
Main Account Dimension List Provider, 660
MainMenu, adding menus, 183. *See also* menus
MaintainUserLicense, 383
managed code
    assemblies, hot-swapping, 84–85
    authoring, 77–84
    third-party assemblies, 76
ManagedHost control, 179–181
manifest, models, 694–695
manual decisions, workflow elements, 253
map element type, defined, 10
map type, reference types, 89
marketing management, element prefix, 23
master data sources, 164–169
master scheduling, performance and, 465
maxOf, X++ select statements, 102
measures, cubes, 321, 333
memory heap, defined, 88
MenuButton, 175
MenuItem
    adding, 182
    AxToolbar, 212
    CopyCallerQuery, 172
    overview, 175
MenuItemButton
    CopyCallerQuery, 172
    overview, 174–175
menus
    adding, overview, 183
    batch jobs, creating, 618
    customizing, 383
    Enterprise Portal, security, 233–235
    form permissions, 360
    Help topic labels, 561
    Help topics, 559–561
    menu definitions, 183
    menu element type, MorphX, 11
    menu item element type, MorphX, 11
    model-driven list pages, creating, 218
    SharePoint navigation, 236
    web, AxActionPanel, 211–212
    workflow artifacts, 266
    workflow menu items, 252, 270
messages
    Enterprise Portal, web parts, 200
    send framework, 411–414
metadata
    access to, Enterprise Portal, 225–226
    associations, 163
    cubes, metadata selection, 311–312, 325–328
    custom service artifacts, 388
    document service, artifacts, 392–393
    element IDs, 692–693
    form data source properties, 168–169
    form metadata, Microsoft Dynamics AX client, 162–164
    Help topic updates, 563
    Help topics, 566–567
    inheritance, 163–164
    layers, 688–690
    managing artifacts, overview, 687–688
    model store API, 703–704
models
    creating, 693–694
    moving to production, 700–703
    overview, 690–692
    preparing for publication, 694–699
    upgrading, 699–700
    providers, workflow, 254–255
    system services, 388
    system services, consuming, 404–407
    X++ collections as data contracts, 391
Metadata service, Enterprise Search, 240–243
MetadataCache, Enterprise Portal, 225–226
meta-model architecture. *See* application meta-model architecture
method invocations, X++ expressions, 95

method overrides, 184–186
method-bound controls, overview, 179
methods, X++ syntax, 118–120
Microsoft Dynamics AX Reporting Project, 282
Microsoft .NET Framework. *See* .NET Framework
Microsoft ASP.NET, processing architecture, 4–6
Microsoft Dynamics AX
  introduction to, 3–4
  model and application database, 7
Microsoft Dynamics AX Application Object Server (AOS). *See* AOS (Microsoft Dynamics AX Application Object Server)
Microsoft Dynamics AX Business Connector
  authoring managed code, 77–84
Microsoft Dynamics AX client
  action controls, 174–176
  Auto variables, 187
  business logic, 188
  control data binding, 173
  control overrides, 172–173
  controls, Design node properties, 173–174
  controls, run-time modifications, 174
  custom lookups, 188–189
  customizing forms, overview, 184
  Excel templates, building, 190–191
  form data sources, 164–169
  form metadata, 162–164
  form patterns, 160–162
  form queries, 170–172
  forms, overview, 159–160
  Help system interaction, 547
  input controls, 178–179
  layout controls, 176–178
  ManagedHost control, 179–181
  method overrides, 184–186
  Microsoft Office client integration, 189–193
  navigation items, overview, 182–183
  overview of, 159
  parts, overview, 181–182
  user templates, adding, 192–193
  Word templates, 191–192
Microsoft Dynamics AX Enterprise Portal. *See* Enterprise Portal
Microsoft Dynamics AX Report Definition Customization Extension (RDCE), 286–288
Microsoft Dynamics AX services
  asynchronous invocations, 409–411
  consuming
    business document updates, 407–409
    external web services, 414
    system services, 404–407
    WCF client sample, 402–404
  custom services, 388–391
  document services
    artifacts, 392–393
    AxdDocument class, 393–394
    AxTable classes, 395
    creating, 395–397
    customizing, 397–399
    overview, 392
  overview of, 385–387
  performance considerations, 415
  publishing, 400–401
  security, 400
  send framework, 411–414
  system services, 387–388
Microsoft Dynamics AX Trace Parser, 479–488
Microsoft Dynamics AX Windows, 3–4
Microsoft Dynamics Public, security, 232–235
Microsoft Excel. *See* Excel
Microsoft Office
  architecture, 4–6
  presentation tier architecture, 9
  processing architecture, 3–4
Microsoft Office client
  Excel templates, 190–191
  integration with, 189–193
Microsoft SharePoint Server. *See* SharePoint Server
Microsoft SQL Server. *See* SQL Server
Microsoft SQL Server Analysis Services. *See* SQL Server Analysis Services (SSAS)
Microsoft SQL Server Reporting Services. *See* SQL Server Reporting Services (SSRS)
Microsoft Technology platform, architecture overview, 4–6
Microsoft Test Manager
  acceptance test driven development (ATDD), 535–536
  ALM solution, 534
  ordered test suites, 539–540
  overview, 533–534
  shared steps, 536–539
  Team Foundation Build, 540–543
  test case evolution, 538
  test selection, 542–544
Microsoft Visio
  Reverse Engineering tool, MorphX, 47–51
Microsoft Visual SourceSafe (VSS), 62–64
Microsoft Visual Studio Integrated Development Environment (IDE), 4–6

Microsoft Visual Studio Team Foundation Server (TFS), 62–64
Microsoft Windows client, architecture, 4–6, 8
Microsoft Windows Server, architecture, 4–6
Microsoft Word client
  architecture, 9
  templates, building, 191–193
migration, EDT Relation Migration tool, 589
minOf, X++ select statements, 102
modal dialog settings, details page, 220–221
Modal, WindowMode, 221
Model Editor, production reports and, 282
Model Manifest, 694–695
model store
  APIs, 703–704
  element IDs, 692–693
  layers, overview, 688–690
  models, creating, 693–694
  models, overview, 690–692
  models, preparing for publication, 694–699
  moving from test to production, 700–703
  overview, 687–688
  upgrading models, 699–700
model-driven list page, creating, 217–219
Model-View-Controller (MVC), 494
Month_LastMonth, time period filters, 345
MorphX
  accounting framework, model element prefixes, 663
  Application Object Tree
    creating elements, 23
    element actions, 25–26
    element layers and models, 26
    modifying elements, 23–25
    navigation, 21–23
    refresh elements, 25
  Best Practices tool, overview, 39–43
  client-side reporting solutions, 276–277
  code compiler tool, overview, 37–39
  Compare tool, overview, 54–59
  Cross-reference tool, overview, 60–61
  datasets, creating, 201
  debugger tool, overview, 43–47
  Enterprise Portal architecture, 196–198
  Enterprise Portal, developing for, 216
  Find tool, overview, 53–54
  Label editor, overview, 33–37
  models, preparing for publication, 694–699
  operations resource framework, element prefixes, 654
  overview of, 6–7, 19–21
  processing architecture, 5, 7–9
  Projects, overview, 27–30
  property sheet, 30–31
  reference element types, 13
  Reverse Engineering tool, overview, 47–51
  source document model element prefixes, 667
  Table Browser tool, overview, 52–53
  tools, overview, 20
  user interface control element types, 11–12
  Version Control tool, overview, 62–71
  X++ code editor, 31–33
MVC (Model-View-Controller), 494

N

Name, metadata property, 169
Named User license, 376–383
namespace, creating service contracts, 389–390
naming conventions
  assemblies, 74–75
  elements, 21–22
  label files, 35
  temporary tables, 584
navigation
  adding navigation items, overview, 182–183
  CreateNavigationPropertyMethods, 591–593
  Enterprise Portal and SharePoint integration, 235–236
  Enterprise Portal, web parts, 200
navigation bar, Enterprise Portal form design, 156–157
navigation layer forms
  design basics, 141–142
  Enterprise Portal design, 156–157
navigation pane, navigation layer forms, 141, 156–157
NetTcpBinding
  publishing services, 401
  system services, consuming, 404–407
NewModal, WindowMode, 221
NewWindow, WindowMode, 221
noFetch, 100
nonExists, 103
NonFatal, exception handling, 231
NotExistJoin, 164
NotInTTS cache, 446–452
NumberSeq
  getNextNumForRefParmId method, 425
Numeric, exception handling, 107

O

object creation operators, X++ expressions, 96
object models, UML, 49–50
object type
    overview, 91–92
    variable declaration syntax, 94
objects
    assemblies, calling managed code, 76
    RunOn properties, 422
observation, events, 521
OCC (optimistic concurrency control), 409
    Microsoft library resources, 577
    update_recordset, 434
OccEnabled, 100
OData feeds, SQL Server Power View reports, 335–340
ODC files, defining, 343
Office client integration applications, development of, 6–9
OLAP (online analytical processing) databases
    cubes, deploying, 303–309
    currency conversion logic, adding to cubes, 328–332
    DSV, overview of, 321
    Report Builder, 346
OnClick
    pop-up controls, 213–215
    validation, 231
OnlyFetchActive, metadata property, 170, 460
OpenPopup, 214–215
operating unit, defined, 634
operational persona, 334–335
operations resource framework
    extensions, 652–653
    MorphX model element prefixes, 654
    overview of, 648–652
    when to use, 652
optimistic concurrency control (OCC), 409
optimistic concurrency, performance, 444–446
optimisticlock
    performance and, 445–446
    select statements, 100
OptionalRecord, 168
organizational model framework
    custom operating units, creating, 639–640
    hierarchy designer, extending, 642
    integration with other frameworks application modules, 637–638
    organization hierarchies, 635–637
    organization types, 634–635
    overview, 634
    scenarios, modeling, 638–639
Origin, element IDs, 692
outer, 103
OuterJoin, joined data sources, 164
overrides, coding for, 184–186, 430

P

pack and unpack methods, 130–131, 202, 617
page definition element type, 15
Page Definition, Enterprise Portal web parts, 200
page rendering, Enterprise Portal architecture, 196–198
page templates, Enterprise Portal and SharePoint integration, 237–239
page title, Enterprise Portal web parts, 200
Page Viewer web part, 336–339
Page.IsValid, 231
pages, Enterprise Portal and SharePoint integration, 237–239
PageTitle, Enterprise Portal web parts, 200
parallel activities
    batch framework overview, 615
    workflow elements, 253
parameter methods, X++ syntax, 129
parentheses, X++ expressions, 96
parm methods, workflow classes, 269–270
Parse, formatting, 230
partitioned tables
    map element type, 10
    QueryFilter, 611–612
partitions
    architecture overview, 4–6
    cubes, deploying, 305–309
parts
    referencing from a form, 182
    types of, 181–182
Pascal casing, 93
PassClrObjectAcrossTiers, exception handling, 107
PastDataAccess, 363
patterns, X++ design and implementation, 128–133
pause statement, X++ syntax, 98
PBA, element prefix, 23
PDB (Program Database) files, third-party assemblies, 73–76
performance
    batch framework and, 466–467, 615
    caching

EntireTable cache, 453–454
overview, 446
record caching, 446–452
RecordViewCache, 454–455
SysGlobalObjectCache and SysGlobalCache, 456
unique index join cache, 452
client/server performance
reducing round-trips, 418–421
write tier-aware code, 422–426
coding patterns
execute X++ code as CIL, 466
firstfast, 476
indexing tips, 475–476
list page optimization, 476
loop iterations, reducing, 477–478
parallel execution, use of, 466–467
patterns for record existence checks, 472–473
repeat queries, avoiding, 473–474
SysOperation framework, 467–472
two queries vs. join, 474–475
compile and running X++ as .NET CIL, 126–128
configuration
AOS configuration, 463–464
client configuration, 464–465
extensive logging, 465
number sequence caching, 465
Server Configuration form, 463–464
SQL Administration form, 462
field lists, 456–462
InMemory temporary tables, 577–582
label file, 230
master scheduling and inventory closing, 465
Microsoft Dynamics AX services, overview, 415
monitoring tools
client access log, 490
database activities, monitoring, 488–489
Microsoft Dynamics AX Trace Parser, 479–488
Server Process ID (SPID), identifying, 489
Visual Studio Profiler, 490–491
overview, 417
restartable jobs and optimistic concurrency, 444–446
surrogate keys, 586
table inheritance and, 598–599
transaction performance
delete_from operator, 435–436
insert_recordset operator, 429–432
overview, 426–427
RecordInsertList and RecordSortedList, 437–439
set-based data operators, overview, 427–428
set-based operations and table hierarchies, 428
transferring code into set-based operations, 439–444
update_recordset operator, 432–435
Unit of Work, 599–601
PerInvocation, extensible data security, 367
period templates, time period filters, 344–345
Periodic, Area Page design, 145
permissions
debugging security, 373–375
form controls, 359
forms, 356–359
privileges, creating, 359–361
security framework overview, 353–356
table permissions, coding, 369–371
PerSession, extensible data security, 367
personas, defined, 334–335
perspective element type, defined, 11
perspectives, metadata, 325–326
pessimisticLock, select statements, 100
pessimisticlock, update_recordset, 434
PivotTables, displaying in Role Centers, 340
placeholders, labels, 36–37
policies
debugging security, 373–375
extensible data security policies, creating, 364–369
extensible data security policies, debugging, 368–369
organization model framework, 637
policy context, defined, 365
policy query, defined, 365
queries, data security policies, 365–369
security framework overview, 353–356
policy modeling, 5
polymorphic associations, 91, 132, 596–598
pop-up controls, AxPopup, 213–215
Popup, window type, 174
PopupClosed, 213–215
ports, publishing services, 401
post event handlers, 522–523
postback
updates, 204
validation, 231

PostBackTrigger, 222
postBuild, 494
postRun, 494
Power View, SQL Server
  reports, displaying, 335–340
  reports, editing, 339–340
PowerPivot, 335–340
pre-event handlers, 522–523
prefix, element naming conventions, 21–22
PreRender, chart reporting override, 296
presentation tier
  architecture, 8–9
  Business Intelligence (BI), 299–300
primary entities, details form, 151
primary table, 365–369
PrimaryIndex
  record caching, 447
  surrogate keys, 586–587
print statement, X++ syntax, 98
Private and Shared Projects, 27
privilege element type, 14
privileges
  creating, 359–361
  security framework overview, 353–356
process cycle element type, 14
process cycles, defined, 355
Process State
  accounting, 662–663
  source document framework, 664–665
processingMode, 412
procurement and sourcing, organization model framework, 638
Prod, element prefix, 23
Producer, eventing, 520
product builder, element prefix, 23
Product Configuration, 646–647
Product Dimensions, 643
Product Master, 643, 646–647
product model framework
  extension of, 647–648
  overview, 643–647
  when to use, 647
Product Variant, 643, 646–647
production, element prefix, 23
Program Database (PDB) files, third-party assemblies, 73–76
project, element prefix, 23, 23
Projects
  authoring managed code, 77–84
  automatic generation of, 27–29
  creating, 27
  development tools for, 29
  project types, 30
  upgrades, Compare tool, 56
Projects tool, 20
property sheet, 30–31
Property sheet tool, 20
providers, workflows, 254–255, 260–261
ProviderView, 203
proxies. *See also* .NET Framework
  authoring managed code, 77–84
  Enterprise Portal, developing for, 226–228, 231–232
  Enterprise Portal, security, 232–235
  working with, overview, 73
public key, assembly names, 74–75
PublicPage, 239
publisher, Help system, 550
publishing services, 400–401
publishing, models, 694–699
Purch, element prefix, 23
Purchase Order form, designing, 153–155
purchase, element prefix, 23
purpose, organizational hierarchy, 636

Q

quality checks, Version control tool, 65
queries
  Axd queries, creating, 395–396
  AxFilter, 209–210
  cues, Role Center page design, 143
  data-aware statements, X++ syntax, 99–104
  dimension framework query data, 658–659
  Enterprise Search, 240–243
  field lists, performance and, 456–462
  form queries, overview, 170–172
  full-text support, 606–607
  Microsoft Office client integration, 190
  performance, indexing tips, 475–476
  policy queries, data security, 365–369
  polymorphic, table inheritance and, 165–167
  query element type, 11
  Query object, form queries, 172
  Query service, Enterprise Search, 240–243
  QueryBuildDataSource, 170–172
  QueryBuildRange, 172
  QueryFilter, 172, 607–612
  QueryRun object, form queries, 172

QueryRunQueryBuildDataSource, 171
repeat queries, avoiding, 473–474
system services, 388, 404–407
table inheritance, 594–599
table relations, 588–593
timeout, 463
two queries vs. join, 474–475
workflow documents, 250
Workflow wizard, 251–252
queues
workflow message queue, 257
workflows, overview, 253–254
Quick links, Enterprise Portal, web parts, 200
QuickLaunch, Enterprise Portal web parts, 200
QuickLaunchDataSource, 235
QuickLink, Role Center, 143
quid
value types, overview, 88
variable declaration syntax, 94

R

RDCE (Microsoft Dynamics AX Report Definition Customization Extension), 286–288
RDL (report design definition), 287–288
read permissions
forms, 356–359
menu items, 360
ReadPermissions, 360
real value types, overview, 88
real variable declaration syntax, 94
RecId, table inheritance storage model, 596
record buffers
InMemory temporary tables, 578–582
run-time temporary tables, 585
record caching, 446–452
record context, Enterprise Portal security, 235
record types
reference types, overview, 89
variable declaration syntax, 94
RecordInsertList, 437–439
record-level security (RLS), 430
RecordSortedList, 437–439
RecordViewCache, 454–455
reference types, 13, 89
References, third-party DLLs, 75–76
reflection
classIdGet system function, 672–673
common type, 91
dictionary API, 676–680
instrinsic functions, 670–671
overview, 669–670
table data API, 673–676
treenodes API, 680–685
typeOf system function, 671–672
RefRecId, 132–133
refresh
cacheAddMethod, 419
elements, 25
RefreshFrequency, 367
RefTableId, 133
regulatory requirements, segregation of duties, 354
RelatedTableRole, 593
relational operators, X++ expressions, 96
relations, table, 588
relationships, entity relationship data model, 51
RELATIONTYPE, 596
Released Product, 645
reliable asynchronous mode, SysOperations, 468
Remote Procedure Call (RPC)
grouping calls, 425
MorphX development environment, 6–7
passing table buffers, 425–426
repeatableRead, 100
reporting. *See also* Role Center pages
AxReportViewer, 216
client-side solutions, 276–277
data processing extensions, 288
default chart format, override, 296
Enterprise Portal charts
binding chart control to dataset, 292
chart development tools, 289
data series binding, 292–294
EP Chart Control, creating, 290–291
mark-up elements, 291–292
overview, 289
execution sequence, overview, 278–279
interactive functions, adding, 294–295
list pages as alternatives, 148–149
Microsoft Dynamics AX extensions, 286–288
model elements for reports, 282–285
overview, 275–276
planning for, 279–281
production reports, overview, 281–282
rendering, MorphX user interface control element types, 11–12
Report Builder reports, 346

reporting *(continued)*
Report Definition Language (RDL), 282
Report Deployment Settings form, 288
report design definition (RDL), 287–288
report element type, MorphX, 12
Report, Enterprise Portal web part, 200
Reporting Services, data connection file (.rsds), 336
Reports, Area Page design, 145
requirements, identifying, 324–325
server-side solutions, 277–278
SQL Server Power View, displaying data, 335–340
SQL Server Reporting Services (SSRS), 3–6, 8
SSRS extensions, 285
troubleshooting, 296–297
Visual Studio tools, 346–349
requirements, element prefix, 23, 23
Requirements, operations resource framework, 650–651
ResetFilter, 209
resource element types, 16, 22
Resource Group, operations resource framework, 648–652
resource modeling, processing architecture, 5
Resource, operations resource framework, 648–652
REST API, Excel report display, 340
restartable jobs, performance, 444–446
restore, refreshing elements, 25
retail channel, defined, 635
retries, batch jobs, 628
retry statement, X++ syntax, 98
return statement, X++ syntax, 98
Reverse Engineering tool, 20, 47–51, 669
reverse, data-aware statements, 100
Role Center pages
designing, 142–144
displaying analytic content
Business Overview and KPI List web parts, 341–345
Excel reports, 340
overview, 333–335
presentation tools, choice of, 335
Report Builder, 346
SQL Server Power View, 335–340
Visual Studio tools, 346–349
Quick links, 200
user profiles, 308–309
role element type, 14
role-based access control, 353–356
role-based security element types, 14
RoleName, 367
RoleProperty, 367
roles, security
assigning privileges and duties, 361–362
assigning to users, 363–364
customization and licensing, 383
debugging, 373–375
security framework overview, 353–356
role-tailored design, overview, 139–140
root data sources, 164–169
root table, creating, 594
RootTableContext, 223–225
Row, AxGridView, 208
RowCount, 104
RPC (Remote Procedure Call)
grouping calls, 425
MorphX development environment, 6–7
passing table buffers, 425–426
rules, Best Practices tool, 40–43
run method
batch class, creating, 616
datasets, Enterprise Portal, 202
RunAsPermission, 371–372
RunBase. *See also* SysOperations
client/server considerations, 516
inheritance, 510–511
overview, 510
pack-unpack pattern, 512–516
performance and, 420–421
property method pattern, 511–512
SysOperation comparison, 495–510, 613
UML object model, 49–50
RunBaseBatch, 495–496, 616–617
runBuf, code access security, 124–126
RunOn, object instantiation, 422
run-time
control modifications, 174
QueryFilter API, 611–612
table relations, 588–593
workflow, 257–260

S

sales and marketing management, element prefix, 23
Sales Order form, designing, 153–155
Sales, element prefix, 23

Sarbanes-Oxley, segregation of duties, 354
Save button, autogeneration, 220
saveData, 653
SaveDataPerPartition, 611
scheduled batch mode, SysOperations, 468
scheduling, batch framework, 614
schema
    AxdDocument class, 393–394
    prebuilt cubes, analytic data, 310
    reusing tables and fields, creating cubes, 333
    XML schema definitions for message envelopes, 410–411
Script icon, X++ Code Editor, 33
scripts
    execute editor script, 32
    running, shortcut keys X++ code editor, 32
    X++ Code Editor, overview, 33
search
    Enterprise Portal, navigation form design, 156–157
    Find tool, overview, 53–54
    incremental, shortcut key, 32
    search bar, navigation layer forms, 141
    Search Configuration Wizard, 241–243
Secure Sockets Layer (SSL), Enterprise Portal, 232–235
security
    assembly names, 74–75
    best practices, overview, 372–373
    code access security (CAS), 124–126, 371–372
    configuration key element types, 16–17
    debugging, 373–375
    Enterprise Portal, developing for, 232–235
    extensible data security policies, creating, 364–369
    hiding report columns, 286–288
    insert_recordset operator, 430
    license code element types, 16–17
    permissions, controls, 359
    permissions, forms, 356–359
    permissions, server methods, 359
    privileges, assigning to security roles, 361–362
    privileges, creating, 359–361
    role-based security element types, 14
    RunBase, 420
    security artifacts, developing, 356–363
    security framework overview, 351–356
    security role assignments, 355
    service operations, overview, 400
    table permissions, coding, 369–371
    valid time state tables, use of, 362–363
    validate security artifacts, 363–364
security policy element type, 14
security roles
    assigning privileges and duties, 361–362
    assigning to users, 363–364
    customization and licensing, 383
    debugging security, 373–375
segregation of duties, 354
select
    data-aware statements, X++ syntax, 99–104
    field lists, performance and, 460
    select query, sample code, 101
select forUpdate, 104
SelectedIndexChanged, 208
selection, shortcut keys, 32
selectionChanged, SysListPageInteractionBase, 218
semicolon, use in X++, 95
Send API, 411–414
Sequence, exception handling, 107
serialization, pack and unpack methods, 130–131
Server Configuration form
    batch server, configuring, 625–626
    performance and, 463–464
Server Configuration Utility, hot-swapping assemblies, 84–85
server license, 376–383
server methods, permissions for, 359
Server Process ID (SPID), 489
server, method modifier, 119
service contracts, creating, 389–390, 392–393
service group element type, 13
service implementation class, 388–391
service management, element prefix, 23
service-oriented architecture, 386
Service References, 387
services. *See* Microsoft Dynamics AX services
services element types, 13
session caching, Enterprise Portal, 223
session disposal, Enterprise Portal, 223
SetAsChanged, AxFilter, 209
set-based data operators
    delete_from operator, 435–436
    InMemory temporary tables, 578
    overview, 427–428
    table hierarchies and, 428
    tranferring code into, tips for, 439–444
    update_recordset operator, 432–435

setButtonEnabled, SysListPageInteractionBase, 218
setButtonVisibility, SysListPageInteractionBase, 219
SetFieldValue, AxPopupField, 214–215
setGridFieldVisibility, SysListPageInteractionBase, 219
SetMenuItemProperties, AxToolbar, 212
setter method, 592
setTmp, run-time temporary tables, 585
setTmpData, 579–582
Setup, Area Page design, 145
Shared Projects, overview of, 27
shared steps, 536–539
SharePoint
  Enterprise Portal, developing for, 216–217
  Enterprise Portal, integration with
    Enterprise Search, 240–243
    site definitions, page templates, and web parts, 237–239
    site navigation, 235–236
    themes, 243
    web part page, import and deploy, 239–240
  KPIs, adding to KPI List web part, 342–344
  reporting, troubleshooting, 297
SharePoint Server
  architecture, 8
  Enterprise Portal architecture and, 196–198
  Power View reports, displaying, 336–339
  processing architecture, 3–7
SharePoint Services server, 336–340
SharePoint web client applications, development of, 6–9
shift operators, X++ expressions, 96
shop floor controls, element prefix, 23
shortcut keys
  debugger tool, 47
  X++ code editor, 32
ShowContextMenu, AxGridView, 208
ShowExpansion, AxGridView, 208
ShowFilter, AxGridView, 208
ShowLink, 236
signatures, digital
  assembly names, 74–75
  models, signing, 696–697
SimpleList template, 162
SimpleListDetails template, 162
site definitions, Enterprise Portal and SharePoint integration, 237–239
skipAosValidation, 431–432, 434, 436
SkipAOSValidationPermission, 371–372
skipDatabaseLog, 431–432, 434, 436
skipDataMethods, 431–432, 434, 436
skipDeleteMethod, 436
skipEvents, 431–432, 434, 436
SMA, element prefix, 23
SMM,element prefix, 23
source code files, 688
Source Code, Titlecase Update tool, 93
source document framework
  MorphX model element prefixes, 667
  overview, 664–665
  when to use, 665
sourcing and procurement, organization model framework, 638
specialized base enumerations, value types, 88
specialized primitive types, value types, 88
SPID (Server Process ID), 489
SPLinkButton, 212
SQL Administration form, performance and, 462
SQL Server
  architecture, 7
  processing architecture, 3–6
SQL Server Analysis Services (SSAS)
  prebuilt BI solutions, implementing, 301–309
  prebuilt projects, modifying, 319–323
  processing architecture, 3–8
  SSAS server, configuring, 302
SQL Server Analysis Services Project Wizard
  cubes, customizing, 311–319
  cubes, generating and deploying, 303–309, 328–333
  currency conversions, 316–317
  deploying projects, 301
SQL Server Power View
  reports, displaying, 335–340
  reports, editing, 339–340
SQL Server Reporting Services (SSRS)
  architecture, 3–8
  AxReportViewer, 216
  client-side reporting solutions, 276–277
  data processing extensions, 288
  default chart format, override, 296
  Enterprise Portal, web parts, 200
  interactive functions, adding, 294–295
  Microsoft Dynamics AX extensions, 286–288
  model elements for reports, 282–285
  processing architecture, 3–6, 8
  production reports, overview, 281–282
  report execution sequence, overview, 278–279
  report solutions, planning for, 279–281
  reporting, overview, 275–276
  server-side reporting solutions, 277–278

SSRS extensions, creating, 285
troubleshooting, reporting framework, 296–297
SqlDataDictionaryPermission, 372
SqlStatementExecutePermission, 372
SSAS (SQL Server Analysis Services)
prebuilt BI solutions, implementing, 301–309
prebuilt BI solutions, modifying, 319–323
SSAS server, configuring, 302
SSL (Secure Sockets Layer), Enterprise Portal security, 232–235
SSRS report element type, MorphX, 12
Standard, TabPage controls, 176–178
Standard, window type, 174
startOperation, 494
Startup Element, debugging managed code, 81–82
state model, workflows, 266–267
StateManager, 266–267
statements, X++ syntax, 96, 99–104. *See also* X++ programming language (code)
static CLR elements, invoking, 109
static file element type, 15
Static Files, Enterprise Portal and SharePoint integration, 237
static RDL reports, 288
Static Report Design, 288
static, method modifier, 119
status bar, navigation layer forms, 142
Storage Dimension Group, 645
str
value types, overview, 88
variable declaration syntax, 94
strategic persona, 334–335
strFmt, 36–37
string concatentation, X++ expressions, 96
style sheet themes, Enterprise Portal and SharePoint integration, 243
Subledger Journal, 660–662
SubMenu, AxToolbar, 212
SubmitToWorkflow, action menu items, 272
SubmitToWorkflow, workflow artifacts, 265
subworkflows, workflow elements, 253
sum, X++ select statements, 102
summary page, Help system, 550–551
SupportInheritance, 594
surrogate foreign keys
CreateNavigationPropertyMethods, 591–593
performance and, 168, 586–587
table relations, 590–591
surrogate keys, overview of, 585–587
SvcConfigUtil, publishing services, 401
switch statement, X++ syntax, 98
synchronization
AxFilter, 209
elements, refreshing, 25
proxies, 82–84
temporary tables, 584
Version control tool, 67–68
synchronization log, viewing, 68
synchronous mode, SysOperations, 468
Sys, element prefix, 23
SysAnyType, 90
SysBPCheck, 42–43
SysBPCheckMemberFunction, 42–43
SysClientAccessLog, 490
SysDatabaseLogPermission, 372
SysEntryPointAttribute, 359, 371, 400
SysGlobalCache, performance and, 456
SysGlobalObjectCache, performance and, 456
SysListPageInteractionBase, 218–219
SysModel, reflection table, 675
SysModelElement, reflection table, 675
SysModelElementData, reflection table, 675
SysModelElementLabel, reflection table, 675
SysModelElementSource, reflection table, 675
SysModelElementType, reflection table, 675
SysModelLayer, reflection table, 676
SysModelManifest, reflection table, 676
SysModelManifestCategory, reflection table, 676
SysOperation
attributes, 495
classes, 494
creating batch-executable class, 616–617
overview, 493–494
RunBase comparision, 495–510
SysOperationAutomaticUIBuilder, 494
SysOperationContractProcessingAttribute, 495
SysOperationController, 495–496
SysOperationDisplayOrderAttribute, 495
SysOperationHelpTextAttribute, 495
SysOperationLabelAttribute, 495
SysOperations
overview, 420, 613
performance, 467–472
SysOperationUIBuilder, 494
SysPackable interface, 495–496
SysQueryForm, timeout settings, 463
SysTableBrowser, 52–53
system documentation element type, 16

system function statement, X++ syntax, 98
system services
    consuming, 404–407
    overview, 387–388
system workflows, defined, 246–249. *See also* workflow
SystemAdministratorHelpOnTheWeb, 549
SystemFatal, exception handling, 232
SystemFilter, 209
SystemManaged, form permissions, 358
SystemTable, 611
SysTest framework, new features, 527–533
SysTestCheckInTestAttribute, 528–533
SysTestFilterStrategyType, 531–533
SysTestInactiveTestAttribute, 528–533
SysTestListenerTRX, 541
SysTestMethodAttributes, 528
SysTestNonCheckInTestAttribute, 528–533
SysTestTargetAttribute, 528
SysVersionControlFileBasedBackEnd interface, 71

T

table. *See also* temporary tables
    buffers, passing by value, 425–426
    buffers, set-based data operators, 428
    document services, customizing, 397
    inheritance, overview, 165–167
    permissions, coding, 369–371
    reference types, overview, 89
    table data, reflection API, 669–670, 673–676
    table hierarchies and set-based operators, 428
    table index, database query sample, 101
    table maps, common type, 91
    table relations, overview, 588
    table-level patterns, 131–133
Table Browser tool, 20, 52–53
table collection element type, defined, 11
table element type, defined, 10
table inheritance
    modeling, 594–595
    performance and, 598–599
    polymorphic behavior, 596–598
    storage model, 596
table of contents, Help system, 550, 563–565
Table References, 589
Table, metadata property, 169
TableContextList, Enterprise Portal, 223–225
TableDataKeys, Enterprise Portal, 224–225
TableOfContents template, 162
TabPage control, 176–178
tabular models, PowerPivot, 336–340
tactical persona, 334–335
TargetClassTest, 528
TargetControl, AxPopup, 215
TargetControlID, AxLookup, 211
TargetId, AxPopupField, 214–215
tasks
    code compiler, 37–39
    logical approval and task workflows, 260–262
    task modeling, performance and, 466–467
    workflow artifacts, 265
    workflow elements, 252
Tax, element prefix, 23
Team Foundation Build, 540–543
team, defined, 635
TempDB
    creating temporary tables, 583–585
    extensible data security constructs, 367
    overview, 582–583
    performance and, 423–424
    transferring code into set-based operations, 442–444
templates
    Dynamics AX Reporting Project, 282
    Enterprise Portal and SharePoint integration, 237–239
    EP Chart Control, 290–291
    Excel, 190–191
    form patterns, 160–162
    Help content, 551–552
    time period filters, 344–345
    user templates, adding, 192–193
    Word, 191–192
    workflow types, 251–252
temporary tables
    creating, 583–585
    EntireTable cache, 453–454
    inMemory, 422–423, 428
    InMemory, 578–582
    insert_recordset operator, 429–432
    TempDB, overview, 582–583
    TempDB, performance and, 423–424
    transferring code into set-based operations, 442–444
Terminal Services, performance, 464
Test Listeners, 541
Test Manager. *See* Microsoft Test Manager

testing
new features, 527–533
test selection, 542–544
Visual Studio tools
acceptance test driven development (ATDD), 535–536
ALM solution, 534
ordered test suites, 539–540
overview, 533–534
shared steps, 536–539
Team Foundation Build, 540–543
test case evolution, 538
testsElementName, 528
TextBox, AxPopupField, 214–215
TFS (Visual Studio Team Foundation Server), 62–64
themes, Enterprise Portal and SharePoint integration, 243
third-party assemblies, .NET Framework and, 73–76
third-party integration applications
development of, 6–9
presentation tier architecture, 9
throw statement
exception handling, 105–106
X++ syntax, 98
time
date-effective framework, 601–606
time period filters, Business Overview web part, 344–345
TimeOfDay, value types, 88
TimeOfDay, variable declaration syntax, 94
valid time state tables, 355, 362–363
Timeout
exception handling, 107
Server Configuration form, 463
title, Enterprise Portal web parts, 200
Titlecase Update tool, 93
TitleDateSource, form properties, 174
TODO tasks, code compiler, 37–39
Toolbar, Enterprise Portal web part, 200, 212–213
topics, Help system
add labels, fields, and menu items, 559–561
context-sensitive topics, 561–562
create in HTML, 552–559
non-HTML topics, creating, 565–567
overview, 549–550
TopNavigationDataSource, 235
Trace Parser, 479–488
Tracking Dimension Group, 645
transaction details form, designing, 153–155
transaction performance
delete_from operator, 435–436
overview, 426–427
set-based data operators, overview, 427–428
update_recordset operator, 432–435
transaction tracking system (TTS), grouping calls, 425
Transact-SQL
tracing statements, 488–489
transferring code into set-based operations, 442–444
TransDate, currency conversion logic, 330–332
Translations, customizing cubes, 315–316
travel and expense, organization model framework, 638
treenodes, reflection API, 670, 680–685
TreeNodeType, 683–685
troubleshooting
Help system, 572–573
reporting framework, 296–297
tracing code, 487–488
Trustworthy Computing, 372–373
try statement, X++ syntax, 98
TTS (transaction tracking system), grouping calls, 425
ttsAbort, 104
InMemory temporary tables, 581–582
ttsBegin, 104
InMemory temporary tables, 581
ttsCommit, 104
InMemory temporary tables, 581
Tutorial_CompareContextProvider, 58–59
Type Hierarchy Browser tool
overview, 20
table inheritance hierarchy, 594–595
type hierarchies, 89–93
Type Hierarchy Context tool, 20, 89–93
type hierarchy, Cross-reference tool, 60
typed data source, element prefix, 22
typeOf system function, reflection, 669, 671–672

U

Unified Modeling Language (UML)
object models, 49–50
Reverse Engineering tool, overview, 47–51
Unified worklist web part, 201, 256
unique index join cache, 452
Unit of Work
form data overview, 167–168
overview of, 599–601

unpack method, 130–131, 203, 617
update method, field lists, 458
update permissions
  forms, 356–359
  menu items, 360
update_recordset
  table hierarchies and, 428
  transaction performance, 104–105, 432–435
  transferring code into set-based operations, 442–444
UpdateConflict, 107
UpdateConflictException, 107
UpdateConflictNotRecovered, 107
UpdateOnPostback, 204
UpdatePanel
  AxContentPanel, 215
  Enterprise Portal, AJAX, 222
UpdatePermissions, 360
updates
  business documents, consuming services, 407–409
  date-effective framework, 605–606
  Help content, 562–563
  User control web part, 201
upgrades
  Compare tool, 56
  Project and, 29
URL
  Excel reports, displaying, 340
  Help system, AOS, 549
  Power View reports, displaying, 336–339
URL web menu item, 218
U.S. Food and Drug Administration regulations, 354
UseIntList, 391
user access
  security framework overview, 351–356
  validate security artifacts, 363–364
user client access license (CAL), overview, 376–383
User control web part
  Enterprise Portal architecture, 196–197
  Enterprise Portal, AJAX, 222
  Enterprise Portal, overview, 201
user experience, designing
  area pages, 144–146
  details form, 150–153
  Enterprise Portal web client experience, 155–157
  list pages, 146–150
  navigation layer forms, 141–142
  overview, 137–139
  Role Center pages, 142–144
  role-tailored design, 139–140
  transaction details forms, 153–155
  user feedback, importance of, 157–158
  work layer forms, 142
user interface
  control element types, MorphX, 11–12
  Enterprise Portal architecture, 196–198
  labels, localization of, 33
  SysOperationUIBuilder, 494
user profiles, deploying cubes, 308–309
user session info service, 388
user-defined class types, 89
UserDocumentation, Help system, 549, 571
UserFilter, 209
users
  creating, 363
  roles, assigning, 363–364
utcDateTime
  relational modeling, 602–603
  value types, overview, 88
  variable declaration syntax, 94
UtilElements, reflection table, 676
UtilIdElements, reflection table, 676
UtilModels, reflection table, 676

V

valdiation
  Validate property, associations, 48–49
valid time state tables, security, 362–363
validation
  aosValidateDelete, 436
  aosValidateInsert, 438–439
  aosValidateRead, 433
  aosValidateUpdate, 433
  AxdDocument class, 394
  cross-table, document services, 393
  document services, customizing, 397–399
  Enterprise Portal, developing for, 231
  registering methods, postRun, 494
  report server validation, troubleshooting, 297
  skipAosValidation, 431–432, 434, 436
  Table Browser tool, 52–53
  table permissions coding, 370–371
  validateByTree, 656
ValidFrom
  date-effective tables, consistency, 604–606

valid time state tables, 362–363
ValidTimeStateFieldType, 601–603
validTimeState, 100
validtimestate key, 602–603
ValidTimeStateFieldType, 362–363, 601–603
ValidTimeStateMode, 604–606
ValidTo, 362–363, 601–606
value stream, defined, 635
value types, 88
ValueFormatter, Enterprise Portal, 230
values, X++ expressions, 95
variables
Auto variables, overview, 187
common type, 91
declarations, X++ syntax, 93–95
expressions, X++ syntax, 95
extended data type, 92–93
object type, 91–92
reference types, overview, 89
value types, overview, 88
X++ syntax, camel casing, 93
variant configuration technology, Product Master, 646–647
Vend, element prefix, 23
vendors, element prefix, 23
Version Control tool
common tasks, 65
create a build, 71
element history, 69
element life cycle, 64–65
integrating with other version control systems, 71
overview, 20, 62–64
pending elements, viewing, 70
revisions, comparing, 70
vertical application domain partition, 4–6
VerticalTabs, TabPage controls, 176–178
view element type, defined, 10
view type, reference types, 89
ViewState, Enterprise Portal, 228–229
ViewUserLicense, 383
Visio, Reverse Engineering tool, 47–51
Visual SourceSafe (VSS), 62–64
Visual Studio
analytic reports, tools for, 346–349
authoring managed code, 77–84
batch jobs, debugging, 630–631
details page, creating, 219–221
Enterprise Portal architecture, 196–198
Enterprise Portal, developing for, 216
model elements for reports, 282–285
prebuilt projects, modifying, 319–323
presentation tier architecture, 8–9
proxies, Enterprise Portal, 226–228
report execution sequence, overview, 278–279
reporting
chart controls, 291–292
client-side solutions, 276–277
data processing extensions, 288
default chart format, override, 296
interactive functions, adding, 294–295
Microsoft Dynamics AX extensions, 286–288
overview, 275–276
production reports, overview, 281–282
report solutions, planning for, 279–281
server-side solutions, 277–278
SSRS extensions, creating, 285
troubleshooting, reporting framework, 296–297
test tools
acceptance test driven development (ATDD), 535–536
ALM solution, 534
ordered test suites, 539–540
overview, 533–534
shared steps, 536–539
Team Foundation Build, 540–543
test case evolution, 538
test selection, 542–544
third-party assemblies, using, 73–76
Visual Studio Integrated Development Environment (IDE)
overview, 7
processing architecture, 4–6
Visual Studio Profiler, performance monitoring, 490–491
Visual Studio Team Foundation Server (TFS), 62–64
VSS (Visual SourceSafe), 62–64

W

warehouse management, element prefix, 23
warehouses, external data integration, 322–323
warnings, compiler
overview, 37–39
suppressing, Best Practices tool, 41–42
WCF (Windows Communication Foundation), 7–8
web client element types, overview, 14–15

web content element type, 15
web control element type, 15
web frameworks, element prefix, 23
web menu
  AxActionPanel, 211–212
  web menu element type, 14
  web menu item type, 14
  workflow menu items, 252
web module element type, 15
web part element type, 15
web part page
  Enterprise Portal and SharePoint integration, 239–240
  Enterprise Portal architecture, 196–197
web parts
  Enterprise Portal and SharePoint integration, 237–239
  Enterprise Portal, overview, 199–201
  Enterprise Portal, security, 233–235
web platforms. *See* Enterprise Portal
web services
  external, consuming, 414
  Microsoft Internet Information Services, 8
Web, element prefix, 23
web.config
  publisher, adding, 569–570
  session disposal and caching, 223
WebLink, SharePoint site navigation, 236
WebMenuName, 211–212
WF (Windows Workflow Foundation), 249–250
while statement, 99, 103
Window Performance Monitor, 482–483
window statement, X++ syntax, 99
WindowMode, 220–221
Windows client applications
  development of, 6–9
  presentation tier architecture, 8
Windows Communication Foundation (WCF), 7–8
Windows Search Service, Help system, 549
Windows Server AppFabric, 223
Windows Workflow Foundation (WF), 249–250
WindowSize, 220–221
WindowType, 174
Wizard wizard, 29
wizards
  Create New Document Service Wizard, 393
  Label Files wizard, 35
  Project development tools, 29
  Search Configuration Wizard, 241–243
  SQL Server Analysis Services Project Wizard, 301, 303–309
  Wizard wizard, 29
  Workflow wizard, 251–252
WKEY, Enterprise Portal security, 235
WMS, element prefix, 23
Word
  architecture, 9
  templates, building, 191–193
work layer forms
  Enterprise Portal design, 157
  overview of, 142
workflow
  activation of, 270–274
  architecture, 256–262
  categories, creating, 268
  creating artifacts, and business logic, 264–265
  display menu item, adding, 270
  document class, creating, 268–270
  elements of, 252–253
  event handlers, 252
  implementation, overview, 263–264
  infrastructure for, 246–249
  logical approval and task workflows, 260–262
  menu items, 252
  overview, 245–246
  providers, 254–255
  queues, 253–254
  state management, 266–267
  Windows Workflow Foundation (WF), overview, 249–250
  work items, 256
  workflow categories, 251
  workflow document and workflow document class, 250
  workflow editor, 255–256
  workflow instances, 256
  workflow life cycle, 262–263
  workflow run time, 257–260
  workflow types, 251–252
Workflow
  activatefromWorkflowConfiguration, 273
  activatefromWorkflowSequenceNumber, 273
  activatefromWorkflowType, 273
workflow approval element type, MorphX, 12
workflow category, workflow artifact, 264
workflow document class, workflow artifacts, 265
workflow document query, workflow artifacts, 265
workflow element types, MorphX, 12–13

workflow provider element type, MorphX, 13
workflow started message, 259–260
workflow task element type, MorphX, 12
workflow type element type, MorphX, 12
workflow type, workflow artifacts, 265
Workflow wizard, 251–252
WorkflowDataSource, 271
WorkflowDocument, 269
WorkflowEnabled, 271
WorkflowType, 271
Workspace, window type, 174
WREC, Enterprise Portal security, 235
write method, 168
write tier-aware code, performance and, 422–426
writing method, 168
written method, 168
WSS (Windows Search Service), 549
WTID, Enterprise Portal security, 235

X

X++ code editor tool
  overview, 20, 31–33
  shortcut keys, 32
X++ collections, data contracts, 391
X++ programming language (code). *See also* MorphX
  assemblies, calling managed code, 76
  attributes, 123–124
  batch jobs, debugging, 630–631
  classes and interfaces, overview, 117–118
  CLR interoperability, 108–112
  code access security (CAS), 124–126
  code element types, 13
  COM interoperability, 112
  comments, syntax for, 115
  compiler, overview, 37–39
  compiling and running X++ as .NET CIL, 126–128
  data-aware statements, 99–104
  date-effective tables, data retrieval, 603–604
  debugger tool, overview, 43–47
  delete_from operator, 435–436
  delgates, 120–122
  design and implementation patterns, 128–133
  dictionary, reflection API, 676–680
  exception handling, 105–108
  executing as CIL, 466
  expressions, 95
  fields, 118
  instrinsic function, reflection, 670–671
  introduction to, 87
  Jobs, 88
  macros, 113–115
  methods, 118–120
  MorphX development environment, 6–7
  pre- and post-event handlers, 122–123
  processing architecture, 5
  proxies, Enterprise Portal, 226–228
  RecordSortedList and RecordInsertList classes, 438–439
  reference types, 89
  referencing labels from, 36–37
  reflection, overview, 669–670
  set-based data operators, overview, 427–428
  set-based data operators, transferring code into, 439–444
  statements, overview, 96
  syntax, overview, 93
  table data, reflection API, 673–676
  table element type, 10
  treenodes, reflection API, 680–685
  troubleshooting, tracing, 487–488
  type hierarchies, 89–93
  type system, 88–93
  update_recordset operator, 432–435
  value types, 88
  variable declarations, 93–95
  XML documentation, 116
xClassTrace, 483
XDS (extensible data security framework)
  debugging data security policies, 368–369
  organization model framework integration, 356
  policies, creating, 364–369
  temporary table constructs, 368
XML documentation
  AxdDocument class, 393–394
  EPSetupParams, 237
  header insert, shortcut key, 32
  Help system, table of contents, 563–565
  schema definitions for message envelopes, 410–411
  system services, consuming, 406–407
  third-party assemblies, using, 73–76
  X++ syntax, 116
XMLHttpRequest, 222
XPO files, 688
Xpp-PrePostArgs, 122–123
xRecord type, reference types, overview, 89