Professional
Microsoft® Search
SharePoint® 2007 and Search Server 2008

Thomas Rizzo
Richard Riley
Shane Young

Wiley Publishing, Inc.
# Professional Microsoft® Search

<table>
<thead>
<tr>
<th>Introduction</th>
<th>xxvii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1: Introduction to Enterprise Search</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2: Overview of Microsoft Enterprise Search Products</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 3: Planning and Deploying an Enterprise Search Solution</td>
<td>27</td>
</tr>
<tr>
<td>Chapter 4: Configuring and Administering Search</td>
<td>45</td>
</tr>
<tr>
<td>Chapter 5: Searching LOB Systems with the BDC</td>
<td>85</td>
</tr>
<tr>
<td>Chapter 6: User Profiles and People Search</td>
<td>131</td>
</tr>
<tr>
<td>Chapter 7: Extending Search with Federation</td>
<td>155</td>
</tr>
<tr>
<td>Chapter 8: Securing Your Search Results</td>
<td>189</td>
</tr>
<tr>
<td>Chapter 9: Customizing the Search Experience</td>
<td>215</td>
</tr>
<tr>
<td>Chapter 10: Understanding and Tuning Relevance</td>
<td>253</td>
</tr>
<tr>
<td>Chapter 11: Building Applications with the Search API and Web Services</td>
<td>289</td>
</tr>
<tr>
<td>Index</td>
<td>333</td>
</tr>
</tbody>
</table>
Professional
Microsoft® Search
For my lovely daughter, Lexi, and my amazing wife, Stacy, this book is dedicated to you, for the sacrifices you made and the support you gave me throughout the process. Also, for her example of strength and courage in the face of fierce adversity, this book is dedicated to Dyana.

—Tom Rizzo

For my incredibly understanding wife, Sarah, and growing bump, thank you for putting up with me over the past few months and not complaining when I’ve been doing, this instead of what I should have been doing; I promise I’ll paint the nursery now!

—Richard Riley
About the Authors

Tom Rizzo is a director in the Microsoft SharePoint product management team. Before joining the SharePoint team, Tom worked in the Microsoft Exchange and SQL Server product management teams. Tom is the author of six development books on a range of Microsoft technologies.

Richard Riley is a senior technical product manager in the Microsoft SharePoint product management team. He is responsible for driving Technical Readiness, both within, and outside of, Microsoft and specializes in Search Server 2008 and the Search features of SharePoint Server 2007. He has more than seven years of experience at Microsoft and has worked as a consultant in Microsoft Consultancy Services, and as a technical specialist in sales. He has over 10 years of industry experience and is a frequent speaker at Microsoft Technical Events.

Shane Young is the owner of SharePoint911. He has over 12 years of experience designing and administering large-scale server farms using Microsoft enterprise technologies. For the past three years, he has been working exclusively with SharePoint products and technologies as a consultant and trainer for www.SharePoint911.com. Shane has been recognized by Microsoft as an authority on SharePoint and is among an elite group of Microsoft Office SharePoint Server 2007 MVPs. Shane also maintains a popular SharePoint focused blog, http://msmvps.com/blogs/shane, which contains a lot of beneficial technical information about SharePoint administration.

About the Technical Editor

Andrew Edney has been an IT professional for more than twelve years and has worked for a range of high-tech companies, including Microsoft, Hewlett-Packard, and Fujitsu Services. He has a wide range of experience in virtually all aspects of Microsoft’s computing solutions, having designed and built large enterprise solutions for government and private-sector customers. Andrew is also a well known speaker and presenter on a wide range of information systems subjects. He has appeared at the annual Microsoft Exchange Conference in Nice. Andrew is currently involved in numerous Microsoft beta programs, including next-generation Windows operating systems and next-generation Microsoft Office products, and he actively participates in all Windows Media Center beta programs. In addition, Andrew has written a number of books, including Windows Home Server User’s Guide (Apress, 2007), Pro LCS: Live Communications Server Administration (Apress, 2007), Getting More from Your Microsoft Xbox 360 (Bernard Babani, 2006), How to Set Up Your Home or Small Business Network (Bernard Babani, 2006), Using Microsoft Windows XP Media Center 2005 (Bernard Babani, 2006), Windows Vista: An Ultimate Guide (Bernard Babani, 2007), PowerPoint 2007 in Easy Steps (Computer Step, 2007), Windows Vista Media Center in Easy Steps (Computer Step, 2007) and Using Ubuntu Linux (Bernard Babani, 2007).
Credits

Acquisitions Editor
Katie Mohr

Development Editor
Christopher J. Rivera

Technical Editor
Andrew Edney

Production Editor
Debra Banninger

Copy Editor
Foxxe Editorial Services

Editorial Manager
Mary Beth Wakefield

Production Manager
Tim Tate

Vice President and Executive Group Publisher
Richard Swadley

Vice President and Executive Publisher
Joseph B. Wikert

Project Coordinator, Cover
Lynsey Stanford

Proofreader
Nancy Carrasco

Indexer
Jack J. Lewis
There are a lot of folks to acknowledge, who helped make this book possible. If I miss anyone, I apologize! First, I want to thank Jim Minatel, Katie Mohr, and Christopher Rivera at Wiley. The three of them made this book possible and also pushed us along in the process at the right times. I also want to thank our production editor Debra Banninger and our technical editor Andrew Edney. Both of them made our words and technical concepts crystal clear. I also want to thank my coauthors who went on this exciting and chaotic journey with me. Finally, I want to thank the SharePoint search team at Microsoft. They are one of the most dedicated teams in delivering high-quality, customer-centric solutions and are always willing to answer questions or provide feedback.

— Tom Rizzo

Writing a book takes much more than one person and a keyboard, and this one is no exception, I’d like to say a huge thank you to the very patient team at Wiley, particularly Katie Mohr and Christopher Rivera, and my coauthors whom I’m sure were all quietly tearing their hair out at my habitual lateness with content (including this page). I’d also like to say a heartfelt thanks to my colleagues in the Search team at Microsoft, whom I’ve repeatedly peppered with questions: Puneet Narula, Keller Smith, Sage Kitamorn, Sid Shah, Dan Blood, Michal Gideoni, Dmitriy Meyerzon, Karen Beattie Massey, Dan Evers, and Brenda Carter. Last, but definitely not least, a thank you to Steve Caravajal, who rescued me from a deep hole with the People Search chapter — I owe you one.

— Richard Riley

I would like to thank the SharePoint MVPs, my friends on the Microsoft product team, and the awesome staff at SharePoint911. I want to send out a special thanks to my wife, Nicola. Without her understanding and support, writing two books at the same time would never have been possible. Also, I have to send a shout out to my two dogs, Tyson and Pugsley. I am sure I missed out on several rounds of throwing the ball while I was busy typing away, but through thick and thin, they lay at my feet. I love you little Sparky!

— Shane Young
# Contents

Introduction xxvii

## Chapter 1: Introduction to Enterprise Search 1

**Why Enterprise Search**
- A Tale of Two Content Types 1
- Security, Security, Security 2
- Algorithms to the Rescue 2
- We All Love the Web and HTTP 3

**Conclusion** 4

## Chapter 2: Overview of Microsoft Enterprise Search Products 5

**Enterprise Search Product Overviews**
- Windows Desktop Search/Windows Vista 5
  - Features in Windows Vista Search 6

**Windows SharePoint Services**
- SharePoint Search Architecture 11
  - Crawling Content 12
  - Searching Content 13
  - Configuring Search 14
  - Platform Services 14

**Microsoft Search Server 2008**
- Simplified Setup and Administration 16
- Federation Capabilities 18
- Different Editions of Search Server 2008 19
- What about WSS and Microsoft Office SharePoint Server? 20

**Microsoft Office SharePoint Server**
- People Search 20
- Business Data Catalog 21
- The Microsoft Filter Pack 23

**Connectors for Documentum and FileNet**

**Windows Live**

**FAST and SharePoint**

**Other Server Products (Exchange, SQL)**

**Conclusion** 25
Contents

Chapter 3: Planning and Deploying an Enterprise Search Solution 27

Key Components 27
The Index Role 27
The Query Role 28
The Shared Services Provider 28
The Database Server 29

Search Topologies 29
Single Server 29
A Small Farm 30
A Three-Server Farm 31
A Medium Server Farm 31
Larger Farms 32

Search Software Boundaries 33

Hardware Sizing Considerations 34
The Index Server 35
Query Servers 37
Database Servers 37
Testing 38
Performance Monitoring 39

Search Backups 41
Index Server Recovery Options 42
Using Federation to scale? 43
Conclusion 44

Chapter 4: Configuring and Administering Search 45

Configuring Search from Central Administration 45
The Search Services 45
The Office SharePoint Server Search Service 46
Windows SharePoint Services Search 50
Manage Search Service 52
Manage Content Database — Search Server 2008 57

Configuring Search from the Shared Services Provider 58
Creating or Editing the SSP Settings 59
SSP Search Administration 59
The Default Content Source 60
Full versus Incremental Crawls 62
Search Schedule 63
Additional Content Sources 64
Interacting with a Content Source 65
Crawl Rules 66
Contents

BDC Web Parts, Lists, and Filters 109
  Business Data List Web Part 110
  Business Data Related List Web Part 110
  Business Data Item Web Part 111
  Business Data Actions Web Part 112
  Business Data Item Builder Web Part 112
  BDC in SharePoint Lists 113

Modifying Your BDC Profile Page 116

Searching the BDC 117
  Adding a Content Source for Crawling BDC Data 117
  Mapping Crawled Properties 119
  Create a Search Scope 119

SharePoint Designer and the BDC 120

The BDC API 122
  The BDC Assemblies 122
  The Microsoft.Office.Server Namespaces 123

Putting It Together: Building Custom Applications for the BDC 123
  Connecting to the Shared Services Database 125
  Displaying LOBSystemInstances 125
  Working with Entities 125
  Working with an Entity – Finders, Fields, and Methods 126
  Executing a Method and Displaying the Results 126
  Working with Associations and Actions 128

TroubleShooting the BDC 129

Conclusion 129

Chapter 6: User Profiles and People Search 131

User Profiles 131
Managing User Profiles 133
  Profile Services Connections 133
  Configuring Profile Imports 134
  Profile Properties 138
  Configuring Profile Properties 139
  BDC Supplemental Properties 142
  Configuring for BDC Import 143

People Search 145
  The Search Center 145
    People Search Page and Tab 146
    Results Page 147
Contents

Customizing People Search 150
    Customizing the Advanced Search Options Pane 151
    Adding Custom Properties to the People Results Page 152
Summary 154

Chapter 7: Extending Search with Federation 155

The Concept 156
Crawl and Index or Federate? 158
    Unified Result Set 158
    Freshness of Results 158
    Avoiding “Double Indexing” Content 159
    Content outside Your Firewall 159
Federated Search Locations 160
    Triggers 163
        Always 164
        Prefix 164
        Pattern 164
            Using Named Capture Groups to Send Only Specific Query Terms 165
    Location Type 168
    Query Template 168
        OpenSearch 1.0/1.1 168
            Required and Optional Query Template Tokens 170
    Search Index on this Server 171
        More Results Link 172
    Display Information 172
        Specifying a Branding Icon 174
        Customizing the Title of a Federated Results Web Part 174
        Writing Custom XSL 175
    Properties 175
    Sample Data 176
Restrictions 177
Credentials 177
    Anonymous 178
    Common 178
    User 178
    Configuring Search Server 2008 for “User” Authentication 179
    Configuring SharePoint Server for User Authentication 180
Security Trimming Federated Results 180

xxi
Contents

Federation Web Parts 182
  Federated Search Results Web Part 183
  Federated Search Results Web Part Properties 185
    Location 185
    Results per Page 186
    Use Location Visualization 186
    Retrieve Results Asynchronously 186
    Results Query Options 186
    Show More Results Link 187
    Appearance Settings 187
    Caching Location Settings 187
  Top Federated Results Web Part 187
    Multiple Locations 187
    Retrieve Results Asynchronously 188
Summary 188

Chapter 8: Securing Your Search Results 189

Security Architecture in SharePoint 189
  Best Bets 191
  Controlling Indexing to Secure Content 191
  What about IRM-Protected Documents? 191

Custom Security Trimmers 191
  Implementing a Custom Security Trimmer 192
  Registering the Custom Security Trimmer 193
  Performance Considerations 193
  Bringing It All Together: Building a Custom Security Trimmer 194
    Getting Started 194
    Signing Your DLL and Adding It to the GAC 198
    Registering your Customer Security Trimmer 198
    Debugging Your Customer Security Trimmer 200

BDC Security Trimming 201
  Authentication and the BDC 201
  BDC and Search 202
  Performance Implications 202
  Writing Your Custom Security Trimmer 202
  Deploying Your BDC Security Trimmer 206
  Debugging Your BDC Security Trimmer 209

Securing Your Search Server 2008 212
  Default Content Access Account 212
  Single Server Deployment 212
  Server Farm Deployment 212

Conclusion 213
Chapter 9: Customizing the Search Experience 215

Describing the Flow of a Typical Search 215
Customizing the Search Center — No Code 218
  Tab User Interface 219
  Thesaurus and Synonyms 222
  Restarting the SharePoint Search Service 224
  Stemming 224
  Customizing People Results 225
  SharePoint Designer Support 225
Customizing your Search Results — XSLT 225
  Stepping through the XSLT 227
    Root Template 227
    Parameter Section 228
    No Search Results 229
    Search Results 230
    Hit Highlighting Template 230
    Display Size Template 230
    Display String Template 231
    Document Collapsing Template 231
    More Results for Fixed Query 231
    Discovered Definitions 231
  Working with SPD to Create Your XSLT 231
  Customizing Hit Highlighting in Search 235
  Adding and Rendering Custom Properties 237
    Adding a New Managed Property 241
    Customizing Core and Advanced Search Results 243
    Customizing People Search and Results with Custom Properties 246
  Using Fixed Queries 251
  Adding Custom Web Parts 251
Search Community Toolkit 252
Conclusion 252

Chapter 10: Understanding and Tuning Relevance 253

What Is Relevance? 253
Built-In Ranking Elements and Algorithms 254
Understanding and Tuning Relevance 254
  Things You Can Change outside of the Ranking Algorithms 255
    Content Management 255
    Language 260
  Understanding Users’ Query Behavior with Query Logs 269
Contents

User Interface 270
High-Confidence Matches 273
Things You Can Change That Directly Affect the Ranking Algorithm 280
Authoritative and Nonauthoritative Pages 280
Property Weighting 282
Global Ranking Parameters 284
Conclusion 287

Chapter 11: Building Applications with the Search API and Web Services 289

Getting Started with References 289
Administration Object Model 289
SearchContext Object 290
Content Sources 291
Working with the Content Class 291
Creating a New Content Source 293
Custom, File Share, Exchange, and Lotus Notes 293
Business Data Catalog Content Source 293
Sample Code: Creating a Content Source 294
Enumerating Existing Content Sources 295
Deleting a Content Source 295
Crawling Content Sources 295
Working with Federated Locations 297
Crawl Rules and File Types 300
Creating a Crawl Rule 300
Working with File Types 302
Scopes 302
Schema Management 304
Querying All Categories 305
Querying All Crawled Properties 305
Querying All Managed Properties 306
Creating and Mapping a New Managed Property 306
Changing Property Weights 307
Editorial Results and Ranking 308
Working with Authoritative Sites and Rankings 310
Search Logs 311
Propagation 312
Writing Search Applications 312
Building Queries 312
Keyword Syntax for Searches 313
URL Syntax 314
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Syntax</td>
<td>315</td>
</tr>
<tr>
<td>Some Example Queries</td>
<td>318</td>
</tr>
<tr>
<td>The Query Object Models</td>
<td>320</td>
</tr>
<tr>
<td>.NET Query Object Model</td>
<td>320</td>
</tr>
<tr>
<td>Web Service Query Object Model</td>
<td>324</td>
</tr>
<tr>
<td>Gotchas When Using the Object Models</td>
<td>330</td>
</tr>
<tr>
<td>Tools to Help You Develop Search Applications</td>
<td>330</td>
</tr>
<tr>
<td>Content Query Web Part and Cross-List Queries</td>
<td>331</td>
</tr>
<tr>
<td><strong>Conclusion</strong></td>
<td><strong>331</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index</th>
<th>333</th>
</tr>
</thead>
</table>
Introduction

Search is a large topic that has grown over the last few years, given the growth of digital information, in both focus (by companies) and importance, for making sure that end users are empowered to find the right information to get their work done. The search landscape and technologies have grown as well, over the same time, making it more difficult to navigate the right technologies and techniques to get the most from your search investments. We were compelled to write this book, since we work with SharePoint and Search everyday, as a way to help others with the changing search landscape.

Who This Book Is For

This book is intended for a range of folks, from the IT administrator to the developer writing search applications. We span many topics in this book to try to cover the breadth of using, administering, and developing on the SharePoint Search technologies. The developer chapters will be important for the administrator to understand, since developers and administrators have to work together to make Search work. On the flip side, the administrator chapters will be important for developers to understand the architecture and administration of Search because, without this knowledge, writing to the APIs will be more difficult. Most readers will benefit from reading all the chapters in this book.

What This Book Covers

This book covers the breadth of the SharePoint Search technologies, from Search Server 2008 to Windows SharePoint Services to Office SharePoint Server. We also include information on the latest search technologies coming from Microsoft, including the new federation capabilities, filter pack, and the recently acquired FAST technologies.

How This Book Is Structured

This book is structured in such a way that you can read it from end to end. The chapters are laid out in such a way that they build on each other, starting with an overview chapter, and ending with an API chapter that shows you how to program against all the technology about which you just learned. If you are new to SharePoint, the first few chapters will be important for you to understand and digest before moving on, since the array of search technologies can be overwhelming for someone new to them. For experienced SharePoint readers, the overview chapters are a good refresher to skim through, but you probably can skip right to the detailed chapters, starting with Chapter 3, Planning and Deploying an Enterprise Search Solution.
Introduction

What You Need to Use This Book
To get the most from this book, you will want a copy of Office SharePoint Server. Windows SharePoint Services or Search Server 2008 will work, but you will not have access to all the search capabilities we talk about in the book. One easy way to get an evaluation copy of SharePoint is to download the SharePoint virtual machine from MSDN. You can find a link to the virtual machine on the SharePoint home page at www.microsoft.com/office/sharepoint. This virtual machine, while large, is preconfigured for you, so that you can start working with the SharePoint Search technologies without having to install all the software and configure it.

Conventions
To help you get the most from the text and keep track of what’s happening, we’ve used a few conventions throughout the book.

Boxes like this one hold important, not-to-be forgotten information that is directly relevant to the surrounding text.

Notes, tips, hints, tricks, and asides to the current discussion are offset and placed in italics like this.

As for styles in the text:

- We show keyboard strokes like this: Ctrl+A.
- We show filenames, URLs, and code within the text like this: persistence.properties.
- We present code in two different ways:
  - We use a monofont type with no highlighting for most code examples.
  - We use gray highlighting to emphasize code that’s particularly important in the present context.

Source Code
As you work through the examples in this book, you may choose either to type in all the code manually or to use the source code files that accompany the book. All of the source code used in this book is available for downloading at www.wrox.com. Once at the site, simply locate the book’s title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book’s detail page to obtain all the source code for the book.

Because many books have similar titles, you may find it easiest to search by ISBN; this book’s ISBN is 978-0-470-27933-5.
Introduction

Once you download the code, just decompress it with your favorite decompression tool. Alternately, you can go to the main Wrox code download page at www.wrox.com/dynamic/books/download.aspx to see the code available for this book and all other Wrox books.

Errata

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, such as a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata you may save another reader hours of frustration, and, at the same time, you will be helping us to provide even higher-quality information.

To find the errata page for this book, go to www.wrox.com and locate the title using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page, you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list including links to each book’s errata is also available at www.wrox.com/misc-pages/booklist.shtml.

If you don’t spot “your” error on the Book Errata page, go to www.wrox.com/contact/techsupport.shtml, and complete the form there to send us the error you have found. We’ll check the information and, if appropriate, post a message to the book’s errata page and fix the problem in subsequent editions of the book.

p2p.wrox.com

For author and peer discussion, join the P2P forums at p2p.wrox.com. The forums are a web-based system for you to post messages relating to Wrox books and related technologies, and interact with other readers and technology users. The forums offer a subscription feature to email you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At http://p2p.wrox.com you will find a number of different forums that will help you, not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to p2p.wrox.com and click the Register link.
2. Read the terms of use, and click Agree.
3. Complete the required information to join, as well as any optional information you wish to provide, and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.

You can read messages in the forums without joining P2P, but in order to post your own messages, you must join.
Introduction

Once you join, you can post new messages and respond to messages other users post. You can read messages at any time on the web. If you would like to have new messages from a particular forum emailed to you, click the Subscribe to this Forum icon, by the forum name, in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works, as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.
Introduction to Enterprise Search

With the explosion of digitally borne information in the workplace, Enterprise Search has become more critical than ever before. Gone are the days when you could remember the location of all the file shares, web sites, and SharePoint sites, where the information you needed was stored. Instead, sites with terabytes of data are normal now, rather than being the anomaly they were just a few years ago. Remembering where you stored something last year, or even last week, has become an exercise in searching for a needle in a haystack. Also, with the growth of Internet Search, companies have begun to question why they do not have as good a search engine inside the firewall as they do outside the firewall. Internal customers are demanding that you provide a robust, scalable infrastructure for them to search against and provide in return relevant and timely results. Not a short order in any way, but reading this book will help!

Why Enterprise Search

Some of you may be scratching your heads, wondering why there is a distinction between Enterprise Search and Internet Search. Aren’t the problem sets and technologies the same between the two? Yes and no. Some of the algorithms and protocols are the same, but some are different. While some Internet technologies grew out of Enterprise Search products, the technologies are distinctly different for a number of reasons that we will discuss.

A Tale of Two Content Types

While there is some overlap between Internet content types and Enterprise Search content types, the majority of the corpuses remain distinct. The Internet is made up mostly of web files ranging from Hypertext Markup Language (HTML) to Extensible Markup Language (XML) with not as much Office document content, while the reverse is true for Enterprise Search, where the majority of content is usually Office documents. Of course, this all depends on the types of content you
Chapter 1: Introduction to Enterprise Search

crawl in the enterprise. With SharePoint and the myriad of content sources it supports, you can get a lot of Office documents, emails, calendars, contacts, people, lines-of-business, or other types of content. This content is very different from web content in that metadata is critical, and the content does not include a lot of linking like web content. Therefore, web-style algorithms will not crack the content effectively, and the results will be less relevant.

Security, Security, Security

As they say, when you are on the Internet, no one knows you are a dog. That is fine for Internet Search, where you do not expect security trimming of results, since the search engine is crawling, indexing, and searching publicly available information. Plus, security is enacted when you try to click on the results in the results list. If you have to log on to access the web site or the file, you will be prompted for your credentials. If you do not have the right credentials, you will be denied access. You will have found the content, but just cannot see it.

Think about the same situation using Enterprise Search. You do not want folks being able to find information they do not have permissions for — not in the search results, not when clicking through to the information, not ever. Imagine that you search your intranet, and you can find the Excel spreadsheet with all the Social Security numbers, salaries, and bonuses for everyone in your organization. The search engine displayed just the first 500 words in the results, but when you click through you get an access denied notification on the spreadsheet. The damage is done. You saw information that was confidential and that should never have been returned to you based on your permissions. This is why security is paramount with Enterprise Search. Information stored inside of a company is usually more restricted and secured than information on the Internet. That is why any good Enterprise Search product has the ability to security trim both at index time and at results time.

In addition, you want your Enterprise Search product to support authentication and authorization against a myriad of authentication and authorization systems, since many environments are heterogeneous. If only one authentication type is supported, the ability to trim based on line of business security, Windows security, or even Lightweight Directory Access Protocol (LDAP) security is compromised and makes the system less useful. It is not uncommon to see three or four authentication and authorization systems inside of companies, while with Internet sites you basically run into forms-based authentication as the primary authentication and authorization method.

We will be covering security, in depth, in this book since it is a broad, diverse, and important topic.

Algorithms to the Rescue

Relevance is the name of the game when it comes to Search, both Internet and Enterprise. To achieve relevance, search algorithms have to scan the corpus of information and rank the results. Different algorithms target different criteria, and then a master algorithm brings together the different rankings to form a single master ranking. The strength of all the different algorithms affects the relevance of the results, and the results affect the value of the search application. We’ll be covering the algorithms used by SharePoint later in the book, but it is a good idea to introduce some of them here, since SharePoint uses a number of different algorithms to try to get you the best result.
Chapter 1: Introduction to Enterprise Search

The main algorithms that SharePoint uses are anchor text, property weighting, property length normalization, URL matching, title extraction, click distance, URL depth, language detection, and file type biasing. The following section quickly describes these algorithms.

- **Anchor Text** — The anchor text is the text or link label that describes a hyperlink. It’s the words you click on for the hyperlink. SharePoint will use the anchor text to establish the rank, but if the query terms are not included in the content of the item that it links to, the content will not be included in the final result list.

- **Property Weighting** — Different properties are more important than others. The property weighting will affect the ranking of the different properties in the final results. You can affect the property weighting by using the search administrative interface or object model. Be careful when changing property weighting, however, since this can severely affect your search results.

- **Property Length Normalization** — Different properties have different lengths. If you have a long property with a lot of values, this could affect relevance, since you can get false positives without taking into account the length of the property. This algorithm can only be customized through the object model.

- **URL Matching** — SharePoint looks at the URL of an item to see if it matches the query terms.

- **Title Extraction** — For Microsoft Office files, SharePoint attempts to extract the title of items, especially if the documents use the default titles from Office such as Slide 1 or Document 1.

- **Click Distance** — The distance between an authoritative source and the target content determines the relevance of the item. SharePoint will lower the rank the further away an item is from an authoritative result.

- **URL Depth** — How far off the main hierarchy content is found affects the relevance ranking of that content. This algorithm calculates the depth and sets the appropriate ranking. Content closer to the root of a site is more relevant than content further down in the hierarchy.

- **Language Detection** — Most users want to find content in their own language, so SharePoint detects the language of the user and ranks content in that language higher than in other languages.

- **File Type Biasing** — Certain types of documents are usually more relevant than others. For example, PowerPoint and Word are more relevant than Excel or plain-text documents. SharePoint ranks certain types of documents higher than other types of documents based on their type.

**We All Love the Web and HTTP**

While different, Enterprise Search and Internet Search share the same love for web standards and protocols. This is where the sharing between the two workloads becomes apparent. There is a significant amount of internal content that is hosted in web standard formats such as HTML, and access to that content is accomplished through the Hypertext Transfer Protocol (HTTP). In addition, the ability to provide the search experience through web-based user interfaces is critical, but Enterprise Search also provides client support for the desktop user, who may want a richer, more interactive experience.

Finally, programming against both types of search engines is similar in that web standards and protocols are used. Many times, both search engines can be customized by just tweaking the URL that you send to the engine and passing different parameters as part of the URL string. In addition, web services and
standards, such as OpenSearch, are critical components for the developer for writing code or customizing the search queries and experience. Sharing these technologies across both environments makes integrating the two technologies easier, which is important since we are seeing more and more convergence between the Internet and intranet experiences.

Conclusion

In this chapter, we took a quick look at Enterprise Search and some of the differences between Enterprise Search and Internet Search. While there are some similarities, there are some glaring differences between the two technologies. Content types, algorithms, and security are the main differentiators. As you read through the rest of the book, keep in mind that these differentiators are critical to your Enterprise Search deployment, relevancy, and development.
Overview of Microsoft Enterprise Search Products

Microsoft has been in the Enterprise Search business for a long time — does anyone remember Index Server and Site Server? Over the years, the Microsoft product line has changed, with the different products evolving and new products being added. When it comes to Enterprise Search, the primary Microsoft products you will deal with are SharePoint and the Windows Desktop Search (WDS) technologies. For completeness though, we will talk about some of the other products that you may encounter when deciding how to build out your Enterprise Search strategy and infrastructure. This includes products such as Live Search and the individual full-text technologies built into Microsoft server products including SQL Server and Exchange Server.

Enterprise Search Product Overviews

As mentioned previously, when evaluating Microsoft Enterprise Search, you will want to look at two primary products — Microsoft Office SharePoint Server and Windows Desktop Search. Many of the search technologies from Microsoft build on the same foundation, so regardless of the product used, the infrastructure and architecture is the same. The following section will outline the different search products from Microsoft, and then we will look at the architecture underlying these products.

Windows Desktop Search/Windows Vista

Given the proliferation of content types and the growth of storage on laptop and desktop machines, finding information on your own PC has become a difficult task. People store information in myriad places even on their own machine, with some storing it in the filesystem, and others using Outlook, and even other places, such as custom application stores. To find all this
information, you need a robust crawl, index, and search engine. With the introduction of Vista and Office 2007, the Windows Desktop Search component has been integrated into the core product so that searches are fast and accurate. For non-Vista users, Microsoft offers Windows Desktop Search as a standalone component.

**Features in Windows Vista Search**

Windows Vista Search includes all the features found in Windows Desktop Search (WDS) 3.01. You get features such as the ability to crawl and index in a number of different formats, including Word, Excel, PowerPoint, OneNote, and Outlook, and the ability to reach out to file servers, SharePoint, and even Live search. Some of the end user features include an integrated search experience with word wheeling (which is sometimes called search as you type), rich previews, and full actions on search results (such as the Windows actions of copying, moving, and deleting, among others), and integration with Office 2007, especially Outlook, to speed searches of email, the calendar, contacts, and tasks. Vista also has Instant Search capabilities, since search is integrated right into the Vista technology and user experience. Finally, if you have ever used Outlook search folders, you will feel right at home using the search folders in Vista. Just search for what you want. You can even save your search as a search folder so that it’s conveniently available to you at any time. Figure 2-1 shows searching in Vista using the integrated Desktop Search technology. Figure 2-2 shows searching in Outlook using the same technology, but in a different user interface. Figure 2-3 shows the word wheeling feature in Vista.
Chapter 2: Overview of Microsoft Enterprise Search Products

Figure 2-2: Searching in Outlook 2007

Figure 2-3: Word wheeling (search as you type) in Vista
Chapter 2: Overview of Microsoft Enterprise Search Products

For IT professionals, Vista and WDS are enterprise capable. They both support 32 and 64 bits, so that you have support for the latest hardware architectures. Both support group policies, and both are deployable via SMS (or, as it’s now called, System Center Configuration Manager!). The index is also obfuscated on the machine, and the results are security trimmed when a user searches. The main thing that an IT pro has to do with the Vista search components is make sure that the right search locations are set up on the user’s workstation. This can be done at deployment time through group policy, or you can come back later and write script code to modify the search locations. Figure 2-4 shows searching a SharePoint site from Vista in the browser.

![Figure 2-4: Searching SharePoint from Vista](image)

Looking under the covers for a moment, to set up the SharePoint search from Vista, you need to modify some policy settings on your machine. If you are not running Vista, you will need to extract the file DesktopSearch.adm from the WDS package by using the /extract command. The best place to find the latest information on how to do this is to search for the *Windows Desktop Search Administration Guide* on the Internet. This guide will have the latest information on the download and any last minute changes to the group policy template documentation.
For Vista users, there is already an existing group policy for Search. To find it, type in `gpedit.msc` in the search box on the Start menu and launch the group policy editor. Remember to click “continue to the user access control popup” when you try to launch the group policy editor. Once it is launched, look under Computer Configuration, Administrative Templates, Windows components and then Search. In there, you will find the different settings to customize search such as index file location, indexing encrypted files, default included and excluded paths, and a number of other settings. You can then change the settings to meet your needs.

To integrate Vista with SharePoint Search, you need to set a different area in the policy settings. Under User Configuration, Administrative Templates, select Windows Components, Instant Search. You need to double-click the Custom Instant Search Internet search provider setting. Once the dialog box comes up, click the Enabled radio button, and then set the string that will show in the Instant Search menu, such as “SharePoint.” Then, in the URL box, enter in the URL to your SharePoint site to perform the search. For example, enter `http://sharepointserver/searchcenter/pages/search_results.aspx?k=%w`. Figure 2-5 shows the setting dialog box. Figure 2-6 shows an Instant Search in Vista after completing the settings.
For Windows XP, you will want to load the group policy template. Once you do that, go into the group policy editor, and, under Computer Configuration, Administrative Templates, Windows Components, Windows Search, there are two settings that you want to modify to have Search work with SharePoint. The first is the Add Primary Intranet Search Location. You need to enable the setting to tell WDS the URL to call to search SharePoint. You will want to format the URL, so that you are calling the search_results.aspx page of SharePoint and passing along the querystring to perform the search you want. In addition, you will want to start the string with the name that you want to call the site. In the most common case, you will just pass along the search terms that Windows gives you in the $w variable. For example, to search a server named MOSS, you would create the URL http://moss/searchcenter/pages/search_results.aspx?k=$w. The complete string would be something like Intranet,http://moss/searchcenter/pages/search_results.aspx?k=$w. You will learn in later chapters all the different parameters that SharePoint takes in the URL. For now, you just need to know that the $k variable represents the keywords that someone wants to search for.

The second setting you can modify to search SharePoint is an optional setting, called Add Secondary Intranet Search Locations. This setting takes the same parameters as the primary intranet setting, but you can delimit multiple locations with a semicolon, such as name,URL1;name,URL2. This setting is good if
you have multiple locations that provide search results, or if you want to allow customized searches (by changing the querystring of the URLs or the search scopes where the searches look for results).

When you finish editing the policies, it is a good idea to force the updates to take effect immediately for testing purposes. To do this, launch a command prompt, and type in `gpupdate /force`. This will apply all the changed policy settings.

**Windows SharePoint Services**

Now that you have seen what the desktop has to offer from a Search standpoint, it is time to turn our attention to the server side. For those of you who have used full-text search in previous versions of Windows SharePoint Services (WSS), before the v3 version, I am sorry. You had to deal with using the SQL full-text search indexing, which is administered separately and also has a separate backup and restore process. With v3, we incorporated the full SharePoint search engine into WSS, so that you would not have to deal with SQL full-text, anymore. It also makes the user experience and upgrade experience between WSS and Microsoft Office SharePoint Server much more seamless.

One caveat, though, is that WSS search is not the same as Microsoft Office SharePoint Server search from a functionality standpoint. WSS search’s main goal is to provide great searches on WSS sites, and not be an Enterprise Search technology. Therefore, you will find new enhancements, such as the ability to crawl and search top level sites and child sites in the WSS product, but you will not find more Enterprise Search types of functionality, such as the ability to crawl non-WSS content. For this reason, WSS is a good starting point for a server-side search, but you will want to use Microsoft Office SharePoint Server or Microsoft Search Server 2008 if you have complex search needs across multiple repositories of data, such as file shares, web sites, SharePoint, Exchange, and Lotus Notes. For those of you who have not heard of Search Server 2008, you will learn about it later in this chapter. It was introduced in March 2008 by Microsoft and is a great way to search WSS in addition to Microsoft Office SharePoint Server.

WSS also differs from Microsoft Office SharePoint Server in that you cannot break out the architecture into roles. Your WSS server performs the roles of both the indexer and the query server, whereas in Microsoft Office SharePoint Server you can have different servers perform these roles. Content in WSS is automatically configured to be crawled, whereas you have much more control over configuration in Microsoft Office SharePoint Server. There are some `stsadm` commands for some administration, but nothing like the administration web site included in Microsoft Office SharePoint Server. Finally, you can crawl multiple content DBs using WSS. The recommendation is to limit the number of content DBs to under 100 for performance and scale reasons.

**SharePoint Search Architecture**

Since WSS and Microsoft Office SharePoint Server share the same architecture, now would be a good time to describe the search architecture for both products, and what is happening under the covers. Search is implemented using the shared service provider architecture in SharePoint. The areas that you need to be aware of in the architecture are the content sources, configuration data, protocol handlers, index engine, IFilters, content index, property store, query engine, end user interface, and platform services. Figure 2-7 shows the architecture that we will walk through in more detail in the next few paragraphs.
Chapter 2: Overview of Microsoft Enterprise Search Products

The first thing you need to have, when you want to search, is content. Content sources define the locations, security, and formats that you want to crawl and index. Out of the box, WSS crawls WSS content, and Microsoft Office SharePoint Server can crawl SharePoint, web sites, file shares, Exchange Server public folders, Lotus Notes, and Documentum and FileNet repositories. With the federation capabilities that were introduced recently in SharePoint, you can extend these content sources to any content source that works with the OpenSearch standard.

Crawling Content

SharePoint crawls a content source by using protocol handlers. The index engine calls the filter daemon to filter the content source, and the filter daemon looks for the correct protocol handler for the content source by looking at the protocol in the start address, such as http: or file:. When the index engine hits the content source and starts crawling the content, the daemon tries to find the appropriate IFilter for that content type. If the right IFilter exists, it scans and indexes the content using the IFilter. The metadata is placed into the property store, as is the ACL information for the item, since SharePoint supports index-time ACL collection. The property store is stored inside of SQL Server.
Chapter 2: Overview of Microsoft Enterprise Search Products

Using the wordbreakers and stemmers, the content of the item is placed into the content index, which is stored in the filesystem. The wordbreakers break apart the content into words and phrases. The engine also looks at the noise word files and filters out the noise words. The stemming component creates all the inflections of the word. Note that stemming only stores in the index the word derivations for certain languages, such as Arabic and Hebrew. For English, stemming is done at query time, not at index time. The indexing engine then creates an inverted index, which makes searches faster than when using a regular index.

You can customize the crawling of your different content sources by using crawl rules. These rules specify the paths to include or exclude and how to follow URLs in the source. Whenever content is crawled, SharePoint creates a crawl log that you can use to troubleshoot crawl issues. This log is the primary way that you will find out why content was not crawled, but the majority of the time you will see that the issue was either physical access to the content, such as the content being unreachable, or incorrect permissions causing the error.

One thing to note is that indexing content is very CPU intensive because of all the calculations that must happen on the content. For this reason, you will want to break apart your index server from your query servers in a query-heavy environment, so that you can scale out your query front ends. Plus, having multiple query servers provides high availability, just in case one of the query servers becomes unavailable. Unfortunately, you can only have a single index server feeding the query servers, so the index server can be a source of downtime if it crashes. Nevertheless, with continuous propagation, which we discuss below, the query servers can continue to function without having the index server available. You will just not get updated content to the query servers from the content sources that the index server was configured to crawl. Also, each SSP can only have one index, but you can have multiple SSPs, and each SSP farm can index up to 50 million documents. If you need to scale beyond 50 million documents, you will need to add farms.

Finally, SharePoint supports continuous propagation of the index from the index servers to the query servers. This propagation allows the query servers to have up-to-date information from the index and provides them with the ability to search a local index, rather than have to query across the network to a network index. Continuous propagation is constantly sending new or changed content, as it is crawled, to your query servers, so you do not have to wait for the entire index to be created before allowing users to be able to search the content.

SharePoint propagates content continuously, creating a master index, and when it crawls, it creates a number of smaller files called shadow indexes. The index server copies these shadow indexes to the query servers. When the shadow indexes reach 10% of the size of the master index, a master merge occurs, where the shadow indexes are merged into the master index, and a new master index is created and propagated.

Searching Content

The main way that folks will search will be the Out of Box (OOB) SharePoint user interface, or your custom application. Either way, the query pipeline will be the same. Once a query is submitted to SharePoint, the query engine takes the search terms and uses the wordbreaker to break apart the terms into individual terms. If stemming is enabled, the engine passes the terms through the stemmer, and all inflections of the search term are generated. All the terms are queried in the property store for matches, and the engine also checks the access control lists (ACLs) of the user and the document in the index.
Chapter 2: Overview of Microsoft Enterprise Search Products

If the user has permissions, the document is returned and ranked with the other returned documents, using the ranking algorithms. If the user does not have permissions for the document, the document is removed from the results. If you have registered a custom security trimmer, the results are trimmed before the results are returned to the user.

If you are using the OOB search interface, a hidden object is created with the results that are shared across the OOB search web parts. The web parts then parse the results and display them in the user interface. As you will learn in later chapters, the search user interface is modular in that different web parts perform different functions. There are web parts that display statistics, keyword matches, the core results, and federation results. You can quickly customize the SharePoint search experience by customizing the properties for these web parts.

Configuring Search

When you configure SharePoint search, the configuration data is stored in SQL Server with the rest of the SharePoint configuration data. This includes the information about your content sources, rules for crawling those content sources, security information about the default content access account, your managed properties and their settings, and a number of other settings. The reason for storing this all in SQL Server is that when you back up your SharePoint information, the search configuration data, the property store, and the index in the filesystem are all backed up and can be restored in the event of a failure.

Microsoft Office SharePoint Server includes a rich configuration user interface as part of its central administration, while WSS uses \stsadm\ commands for configuration. You will see the configuration user interface in future chapters, but be aware that WSS provides a limited configuration environment for your search needs.

Platform Services

While SharePoint provides a rich OOB experience for searching, there will always be scenarios where you want to extend the product to add new capabilities or use the capabilities in your own application. For this reason, SharePoint provides extensibility for the Search technologies through a number of methods. For example, you can write code to administer the server using the server administration object model, or you can write code to perform queries and get search results. The following section outlines the different platform services that SharePoint provides; we will cover these in more detail in later chapters.

Administration Object Model

Using the same object model that Central Administration uses, you can programmatically administer your SharePoint search configuration. The object model allows you to work with content sources, metadata and properties, scopes, keywords and best bets, ranking and relevance, propagation, and the crawl log. There are some operations that are only available through the administration object model, such as changing the weighting on your managed properties. You will want to get very familiar with this object model even if you are only writing search applications. You will use this API to create your content sources and properties and to crawl your content.
Chapter 2: Overview of Microsoft Enterprise Search Products

Query Object Model and Web Services
When you want to write your own search applications, by either extending the OOB functionality or by creating new applications, you will work directly with the query object model or search web services. You may be wondering why there are two APIs. The primary difference between the two is where your code runs. If you are running on the SharePoint server, you will want to use the query object model. This object model is not remotable, so you cannot use it off the SharePoint server. If you want to work with the object model remotely, you will need to use the Query web service, which can be called off the server and uses the standard web services’ APIs. Be aware that the Query web service will need to know whether you are working with WSS or Microsoft Office SharePoint Server, since the web services for the two products are in different locations.

Protocol Handlers and IFilters
To understand how to crawl and index new sources of content, SharePoint uses protocol handlers and IFilters. Protocol handlers tell SharePoint how to access a particular content source, since they provide an enumeration of all the items in the content source. Once the enumeration is passed back to SharePoint, SharePoint determines the type of content and loads the appropriate IFilter. IFilters are components that know how to extract the content and properties from an item. SharePoint ships with a number of protocol handlers, such as an HTTP, SharePoint, Exchange, and file protocol handlers, and IFilters that can work with the Office document formats. You can write your own protocol handlers and IFilters, but if you are running Microsoft Office SharePoint Server, you may want to look at the Business Data Catalog, discussed next.

Business Data Catalog
For many folks, protocol handler and IFilters are too complex. They require a deep knowledge of the Component Object Model (COM), and they are not .NET based. With that in mind, the SharePoint team built a new component in the 2007 release, called the Business Data Catalog (BDC), that allows you not only to crawl and index custom systems, but also to use that data inside of your lists and web parts. The easiest way to think about the BDC is that it is a metadata mapper and transport provider. You tell the BDC where your datasource is located; it supports both ADO.NET and web services, how to log on to that system, how to enumerate the data, and how the data coming from that system maps into SharePoint data structures. From there, you can crawl and index your new BDC datasource as you would any content source that SharePoint supports. Once you have the data from your BDC datasource, you can use that data in lists, web parts, search results, user profiles, or other places in SharePoint. The BDC also works with single sign-on (SSO) to access the back-end system securely and supports security trimming by using either user filters or a results-time security trimmer. There are two “gotchas” with the BDC, however. First, the BDC does not index the content of items, only metadata. So, if you are going to crawl unstructured information, such as files, or content that has data in its body rather than properties, the BDC will not index that data. Second, the BDC is only available in the Enterprise Edition of Microsoft Office SharePoint Server. If you don’t own the Enterprise Edition, you do not own the BDC.
Chapter 2: Overview of Microsoft Enterprise Search Products

Federation Connectors

Recently introduced by Microsoft, federation connectors allow SharePoint to call a remote system to send a query and retrieve the results. Then, SharePoint formats and displays the results in its user interface. SharePoint does not crawl or index the content itself, so your performance will be dependent on the ability of the remote system to process queries and return results quickly. SharePoint supports the OpenSearch standard, which allows you to send a query to other engines that support the standard and get back a result either as Really Simple Syndication (RSS) or Atom Syndication Format (ATOM). The OpenSearch standard was originally created by A9, an Amazon.com company, and is now supported by a number of different search engines. SharePoint extends OpenSearch a bit, as we will learn in later chapters, in order to provide additional functionality.

Custom Security Trimmers

As you learned earlier, certain SharePoint protocol handlers store and use ACLs in the property store. When a user queries for a particular search term, the results are filtered at query time, and SharePoint does not have to go back to the source system to retrieve the ACLs. This works well if your data is contained in one of the datasources in which SharePoint stores ACLs. For example, SharePoint does not store the ACLs for web-based content. If you wanted to filter web content, you would have to have SharePoint logon to the web site and then filter the results before the results are rendered to the user. This is where custom security trimmers come into play. Before the results are displayed to the user or application that generated the query, SharePoint allows you to plug in your own code to security trim the results. You control which content gets returned and which authentication systems you use.

Microsoft Search Server 2008

Microsoft Search Server 2008 is a new addition to the extended SharePoint family. Some of you may know this product under its previous name, Microsoft Office SharePoint Server for Search. Yes, we really did give a product a name that long, especially when you tack on the Standard Edition or Enterprise Edition monikers! To help simplify the naming, as well as the product, we introduced an interim release between the 2007 release and the next release, codenamed Office 14. The new release focuses in on three key areas: simplified setup and administration, federation, and improved security. Let’s take a quick look at the first two of these enhancements.

Simplified Setup and Administration

No one wants to take a day setting up a new search server. With Search Server 2008, you can get a server up and running in under an hour. Search Server 2008 will download and install the prerequisites for you, such as SP1 for Windows Server 2003 and the .NET Framework 3.0. In addition, Search Server 2008 will make sure that you have IIS and ASP.NET installed and that your application pools are correctly configured. Since SharePoint and Search Server 2008 build on top of a complex stack of Windows, SQL and .NET/IIS, these automated configuration and installation steps definitely help remove some of the most common errors when setting up a server.
Chapter 2: Overview of Microsoft Enterprise Search Products

If you use the basic install of Search Server 2008, Search Server 2008 will install SQL Server Express, create and configure all the service and database accounts, provision a shared service provider and configure it, and create a search center site. Again, lots of folks can get SharePoint installed, but there are a number of steps involved in actually getting to your search page and performing a search. The simplified setup makes it a lot easier. Figure 2-8 shows the simplified setup screen.

Once you are up and running, configuring the search server is very easy. Search Server 2008 introduces a new administration screen where you can perform all of your configuration in one spot. Previously, you would have had to go to three or four different sites to perform the same functions. From the administration dashboard, you can view your server status, including system status, active crawl status, and results of recently complete crawls. The dashboard also includes links to adding content sources, creating crawl rules, working with queries and results, getting usage reports, accessing your search center site, and viewing how to help content. Figure 2-9 shows the new administration dashboard.
Chapter 2: Overview of Microsoft Enterprise Search Products

Federation Capabilities

Search Server 2008 introduces new federation capabilities that allow you to have SharePoint send queries to other systems, retrieve and format the results, and display them within SharePoint. The federation capabilities support the OpenSearch standard, which takes the query as a parameterized URL and passes it to the target system. The target system runs the query and returns the results as XML. With the introduction of this new federation capability, Microsoft also introduced some OOB federation providers. The two that come with Search Server 2008 support federating to local content and federating to Microsoft Live Search. You can use the local content source if you want to target searches to various content sources, or if you want to filter in different ways from the main search results. If you want to add other federation locations, you must configure Search Server 2008 to know about the new location. You can import and export federation locations as .FLD files.

To implement the new federation capabilities, SharePoint introduces two new web parts which are the Federated Results web part and the Top Federated Results web part. The Federated Results web part, as the name suggests, displays the results from your federated query source. The Top Federated Results displays the top results, from your different federation sources by default, above the Core Search Results web part. Figure 2-10 shows the new web parts in action.
One thing you should realize about federation connectors is that the results are not ranked by relevance to the core SharePoint results. This means that your federation results will not be interspersed with the results coming back from SharePoint. This is different from using BDC to index and search your content, since BDC content can be interspersed with other SharePoint results and SharePoint will rank the BDC content by its relevance to the rest of the search results. In fact, if you use multiple federation sources, each federation source will be ranked according to whatever algorithm that the federation system providing the search results uses.

**Different Editions of Search Server 2008**

Search Server 2008 comes in two different editions. The first is the Microsoft Search Server 2008 Express version. The Search Server 2008 Express version is free of charge, so you can use it anywhere you like, on your intranet, on the Internet, or anywhere in between. There is a purchasable version of Search Server 2008 called just Microsoft Search Server 2008. The main difference between the Search Server 2008 Express and Search Server 2008 versions is that Search Server 2008 Express is limited to only a single server, so you cannot put together multiple Search Server 2008 Express servers in a farm or have two for disaster recovery and high availability. You could put together multiple distinct Search Server 2008 Express servers, but you will lose the central administration and ease of use.
Chapter 2: Overview of Microsoft Enterprise Search Products

Search Server 2008 also does not have all the features of Microsoft Office SharePoint Server. For example, you do not get People or BDC search with Search Server 2008. You could implement a simple people search using Active Directory, but you would not get the same experience as you would with Microsoft Office SharePoint Server. The same goes for searching Line of Business (LOB) systems in that you could federate to those systems, but you would not get the BDC experience.

**What about WSS and Microsoft Office SharePoint Server?**

For WSS customers, Search Server 2008 can extend WSS with better search capabilities. You can replace your default search setup with Search Server 2008. When you do that, WSS will be able to search across sites, not just in the site and subsites below the current site. Since Search Server 2008 builds on WSS v3, you can do an in-place upgrade of WSS to Search Server 2008 to get this additional search capability. If you do a clean install of Search Server 2008, Search Server 2008 will install WSS v3, and WSS v3 will automatically have the cross-site search capability.

For Microsoft Office SharePoint Server customers, you will need to install the add-on pack for Microsoft Office SharePoint Server to get the new capabilities found in Search Server 2008 on your Microsoft Office SharePoint Server servers. Once you do that, you will get the federation and simplified administration capabilities. As of the time of writing, the pack was not out, but was expected in the summer 2008.

**Microsoft Office SharePoint Server**

Microsoft Office SharePoint Server provides the most comprehensive search experience of all the products we have looked at in this chapter. With Microsoft Office SharePoint Server, you get all the capabilities we talked about in WSS and Search Server 2008. In addition, you get rich people search and rich line-of-business search through the BDC. Like Search Server 2008, Microsoft Office SharePoint Server does differentiate between different editions. Microsoft Office SharePoint Server comes in two flavors: Standard and Enterprise. The Standard Edition provides all the unstructured and people search capabilities, while the Enterprise Edition has the BDC functionality.

**People Search**

Often, when people are searching, they do not want to search for just content but also for people and expertise. With SharePoint, you can search for both. When a search term comes to the SharePoint search engine, the index also contains information about people contained in the user profile store and users’ My Sites. The search center has a scope to search people and display the results using a people-centric user interface rather than a data-centric view. Figure 2-11 shows searching for people in SharePoint.
There are a couple of things you should be aware of with people search. First, you can sort by social distance or by relevance. Social distance will rank the results according to your social closeness to the people results based on your colleague list. SharePoint automatically generates suggested colleagues for you to add to your list by looking at organization reporting structures, and if you’re running Outlook 2007, by using a COM add-on that watches whom you email, and who emails you, and the frequency of those emails. If the person is your colleague, they will appear in the colleague group of results, if they are a colleague’s colleague, then they appear in that group; otherwise, they appear in the everyone else group. If you sort by relevance, you will get a flat list based on the relevance of your search terms.

SharePoint people search also allows for faceted navigation. You can filter your results by job title or by department, so you can pinpoint who you are looking for. This is different from the standard search, which does not provide this capability.

**Business Data Catalog**

As you learned earlier in this chapter, the Business Data Catalog allows you to connect to systems that expose ADO.NET or web services interfaces. With the BDC, you can access these systems and pull data from the back end. Depending on which components of the BDC you use, some of the data will be pulled from the search index, while other data will be pulled real time from the back-end system.
Chapter 2: Overview of Microsoft Enterprise Search Products

For example, when you perform a search, the search engine queries the data and returns results contained in its index. If you click the results and go to a page that uses the BDC web parts, the BDC web parts pull real-time data from the back-end system. Figure 2-12 shows searching for and retrieving BDC data as part of a search.

You can use the BDC data as columns inside a list or have the web parts hosted in your SharePoint pages. The BDC has a notion of a profile page for your business entities. SharePoint makes it easy to customize these profile pages, since they are standard SharePoint web part pages. Figure 2-13 shows a BDC profile page.
Chapter 2: Overview of Microsoft Enterprise Search Products

The Microsoft Filter Pack

One recently released technology from Microsoft is the Microsoft filter pack. The pack extends all the Microsoft technologies that build on the underlying Microsoft Search architecture, such as Vista, WDS, SharePoint, Exchange, SQL, and Search Server 2008. The pack contains IFilters for the Metro formats (.docx, .pptx, .xlsx, etc.), which products other than SharePoint did not support; the ZIP file format; OneNote; and Visio. By installing the pack, you can crawl and search these file types using these new IFilters.

Connectors for Documentum and FileNet

Microsoft recently announced that it was releasing connectors for Documentum and FileNet. These connectors are protocol handlers that allow you to crawl and search Documentum and FileNet repositories. You could implement something similar using federation connectors, but you would not be indexing the Documentum or FileNet content. Instead, you would be depending on the search engine in those two products to perform the search and return the results. These connectors are available with any version of Microsoft SharePoint technologies, including Search Server 2008 and Microsoft Office SharePoint Server.
Windows Live

Lots of folks ask about the difference between SharePoint and Windows Live. Many wonder whether they are built on the same technology, or whether they are distinct technologies. The real answer is: It depends. Many ideas and sometimes implementations are shared between the two technologies, but Internet Search is different enough from the type of search that SharePoint performs that you will never see Windows Live built on SharePoint search technologies or vice versa. Each is a specialized search engine targeted towards its customer needs.

With that said, there is no reason why you cannot integrate the two technologies. Search Server 2008 ships with a Windows Live federated search provider in the box, so that you can get Internet results right in your Search Server 2008 user interface. Windows Live supports taking a parameterized search URL and returning RSS feeds as search results, which makes writing a federation provider easy. Figure 2-14 shows getting back Live results as well as results from other Internet search engines besides Live.

![Figure 2-14: Returning Internet Search results from Live and other sites](image-url)
Chapter 2: Overview of Microsoft Enterprise Search Products

FAST and SharePoint
Microsoft acquired Fast Technology and Transfer (FAST) in April 2008. FAST provides high-end search solutions for commerce sites, such as Best Buy and Enterprise Search technologies. The FAST technologies, which are now part of Microsoft’s Enterprise Search lineup, fit at the high end of the search spectrum. If you think about search as a continuum, Search Server 2008 and Search Server 2008 Express are at the low end of the spectrum, since they are good products but have less functionality than the rest of the Microsoft search products. In the middle is Microsoft Office SharePoint Server, since it provides very good enterprise search capabilities that include unstructured, structured, and people search. At the very high end with a very extensive feature set, such as faceted navigation and customizable search pipelines, are the FAST technologies. You can use all of these technologies together in one environment. For example, you may need a better WSS search capability, so you install Search Server 2008 with WSS, you add Microsoft Office SharePoint Server for your Enterprise Search, and finally you use FAST for your Internet-facing site. If you are looking for high-end search capabilities, you should take a look at the new Microsoft FAST solution.

Other Server Products (Exchange, SQL)
This last section will cover some of the other server products that have full-text search capabilities, namely, Exchange Server and SQL Server. Those two products build on the same infrastructure as the SharePoint search technology, so IFilters will work across all three search engines. The main difference among them is that the Exchange and SQL server teams at Microsoft take the search infrastructure that the SharePoint team builds and extends it for their own purposes. For example, the SQL Server team needs to build technology on top of the search infrastructure to make sure that the full-text search in SQL understands the nuances of searching SQL Server data. In addition, these other groups have to make sure that their backup and restore technologies and administration user interfaces work with their implementation of the search engine. The main difference between SharePoint and the other server technologies is that SharePoint is an Enterprise Search product that searches across many datasources. The full-text search in these other products is product specific and cannot crawl other repositories.

Conclusion
In this chapter, you took a look at the different search technologies from Microsoft and also learned more about the SharePoint search architecture. Microsoft is quickly innovating in the search space and releasing new technologies at a faster pace than its traditional release cycles. Understanding when to use the right search technology will make the difference between a successful project and an unsuccessful one. Make sure to pick the right Microsoft technology for the scenario that you are trying to address. One great thing is that the SharePoint family of products support upgrading, so you can start with Search Server 2008, for example, and move up to Microsoft Office SharePoint Server over time.
Planning and Deploying an Enterprise Search Solution

Are you excited? You should be. If you are reading this chapter, that means you are preparing to deploy the greatest Enterprise Search platform yet built (in these authors’ humble opinions). Perhaps you have already deployed Search? You are still in luck; this chapter has some powerful information on optimization and monitoring.

This chapter will dive into the underpinnings of Search to reveal what makes it so powerful and how to make it run like a high-performance sports car. We start with the foundation of the key components of Search and an explanation of how to allocate those items to various servers to achieve scale and high availability. You will walk through hardware sizing and storage space requirements for the physical servers. Once you have all of the pieces put together, you are ready to use some disk optimization techniques and performance counters to find bottlenecks. Finally, you’ll take a quick trip through backup/restore considerations as they pertain to the search components.

Key Components

The SharePoint Search engine scales from the smallest of deployments to 50 million documents with no problem. That is not to say you can just plug the server in and index something that large. You need to take a look at the pieces that make up Search, and how they can be scaled up and out to achieve the necessary performance. The four key components are: the index and query roles (which are defined by the Office SharePoint Search service), the Shared Services Provider (SSP), and the database server.

The Index Role

This role is responsible for connecting to content, downloading it to the server, capturing the properties (metadata) and security information and writing them to search property store database, word breaking the content (when available) and storing the content in the index on the
file system, and finally propagating that index to any query servers (if necessary). From that giant run-on sentence, it is easy to see this is a very intensive role.

When planning the index role, it is important to realize that with the index server it is necessary to scale up instead of out. Why? You cannot spread the indexing load for an index across multiple servers. Each index file (corresponding to each SSP) can only be created and maintained by one index server. The index server hardware will need to be planned, accordingly (as discussed later in the chapter).

Because each index is assigned to only one index server, that server could be considered a single point of failure in your high-availability scenarios. Fortunately, if you have separated your query role and index role and placed them on different servers, a failure of the index server does not immediately affect the users. The query server will have its own copy of the index for serving search requests, and the search property store database resides on the database server. Your users will still be able to use Search, but you will not have the ability to index new or changed content until you have restored your index server. See the backup section later in the chapter for some options for quickly recovering the index.

The Query Role

This role is responsible for serving the actual user queries. When a user goes to a SharePoint site and inputs a query in the search box, the information is sent to one of the query servers. The query server processes the request against the physical index and the search property store database, returning all possible results. From this result set, the security trimmer then removes any results the user is not authorized to see and returns the results to the web front end server, to be rendered for the user.

The query role can be scaled out, meaning that it can be assigned to multiple servers. Although there is no hard limit to the number of servers, seven or eight seems to be where performance gains stop. Scaling out can be done to achieve high availability or to increase performance. Since a copy of the index is propagated to each query server independently, if one server goes down, the others can continue to respond to search requests. Also, you do not need to provide any load-balancing mechanism if you are using multiple query servers. SharePoint will handle the routing of the user request.

The Shared Services Provider

The SSP is a set of centrally managed, reusable services that are consumed by the various SharePoint web applications. One of these key services is Search. When you create an SSP, you assign it to one (and only one) Index server. That server will be responsible for indexing the content sources defined in the SSP, creating an index that all of your query servers will use to respond to requests. Also, at SSP creation time, you will specify the database server that hosts the search property store database and the name of that database. Additionally, the SSP has extensive settings to help you to manage the behaviors of crawls. Administering these settings is covered extensively in Chapter 4.

The fact that an SSP can only be assigned to one Index server is very important. Unlike in the previous version of SharePoint Search, you cannot use multiple index files. Each SSP has only one index file. Also, even though it is outside the scope of this book, in general, most companies will strive to have only one SSP. If this is the case, then having additional index servers in the farm would be a waste of resources. We will revisit the multiple server topic in the next section.
The Database Server

This is another very crucial piece in the search puzzle. As you already know, SharePoint stores virtually everything on the database server. Upload a file to SharePoint, and it is stored in the content database. Import users from Active Directory, and they are stored in the database. Search is no different. As noted above, during the indexing process, information is written to the search property store database, and then the query role uses that database to give results to the user. Search is very reliant on the database server, and, therefore, you should plan accordingly.

In almost every environment, the Search performance bottleneck is the property store database. Later in the chapter we will discuss options for overcoming this bottleneck. Also, you should plan your high-availability needs for the property store database. This is a single point of failure in your search hierarchy and should be planned for accordingly. There are many different options for making a SQL database highly available.

Search Topologies

Now that you have an understanding of the various components that are crucial to Search, let’s examine how to spread those across various numbers of servers. We will look at how we can do this to achieve greater performance and high availability.

In this section as you review the various topologies, you will notice the assignment of the WFE role. This is the web front end role that is responsible for rendering the SharePoint pages to the user. We will not be discussing the performance or scale considerations of this component in this book, but it is important to understand that this server plays two roles from a search perspective. First, this role is responsible for accepting user search input and displaying the query results to them. Second, in order for the index server to crawl the SharePoint content, it must be connected to a WFE server.

For more information on how to configure Search in the topologies described below, see Chapter 4. Look for the section on the Office SharePoint Server Search service. There, you will find information on how to assign the various roles, what accounts to use, and how to change the performance setting.

In all of the pictures that follow, the database Server is represented by a single server. The truth is SharePoint does not know or care what the database server portion of the topology is; it just wants you to have a supported Microsoft SQL Server (2000 or 2005) configuration. So, in any of the following diagrams, you could really have a single SQL Server, a SQL cluster, a SQL Mirror set, or any other supported SQL Server topology. SharePoint just calls the database server name you provide and asks for the database name. If SQL responds, that is all that SharePoint cares about.

Single Server

This is the simplest of environments. In this setup, you have one server hosting everything SharePoint (see Figure 3-1).
You configure the Office SharePoint Server Search service to host both the index and query role on this one server. This means that, when the physical index is created, it will not need to be copied to any additional server (propagated). SharePoint will use the one index file both for writing updates and for serving queries, keeping you from having to store multiple copies of the file. Of course, from a High Availability (HA) perspective, you are not HA. If this server is offline all of SharePoint is down, including Search.

**A Small Farm**

This environment is exactly like the single-server setup, except that the database portion has been offloaded to its own server (see Figure 3-2).

For Search, everything is as shown previously from the index file perspective. What has changed is that the search property store database is on a different server. Removing the IO burden of SQL Server to a different box allows for increased performance across the board, although it does not allow HA.
Chapter 3: Planning and Deploying an Enterprise Search Solution

A Three-Server Farm

This is not a very typical setup because of the lack of HA, but it does provide some good discussion points. In this setup, you have a server dedicated to the web and query roles, and one dedicated to the index. The database has been offloaded (see Figure 3-3).

![Figure 3-3: A three-server farm](image)

When you tell the index process to begin crawling the SharePoint sites hosted in this farm, SERVER2 will have to connect to SERVER1 to request the pages. SERVER2 will then update the index file located on its local file system. As the index file is updated, it will have to propagate the index to SERVER1, so that it can respond to queries. Also, SERVER2 will write to the property store database that is hosted on DBSERVER.

One option to consider would be to add the WFE role to SERVER2. Then, you could tell SERVER2 to target itself for crawling content (options for doing this are discussed in Chapter 4), thus reducing the burden on SERVER1, which is responsible for serving pages to the users and reducing the amount of network traffic. This is a recommended best practice for performance reasons.

This setup has still not achieved HA. Looking at the diagram, many administrators will ask, “why not set up SERVER2 to be query server also?” This is a good question. Luckily, there is also a good answer. When SERVER2 updates its index file, it will then check to see whether it is also a query server. If it is not, then it will find all query servers in the farm and propagate to them a copy of the index, so that they can respond to search queries. Now, what happens when SERVER2 checks to see whether it is a query server and discovers that it is? At that point, the server will assume that there is no other query server in the farm, so it will not propagate the index. SERVER1 never gets a copy of the index. When users perform searches that try to use SERVER1 as the query server, they will get no results.

A Medium Server Farm

At last, your first look at HA. In this setup, we took the previous farm and introduced SERVER3, which is a mirror image of SERVER1. We have also changed DBSERVER to represent a database cluster. Now, we have no single point of failure. When SERVER2 builds its index, it will propagate it to both SERVER1 and...
Chapter 3: Planning and Deploying an Enterprise Search Solution

SERVER3. If either SERVER1 or SERVER3 goes down, the other one can continue to serve web pages and respond to search queries. If DBSERVER goes down, it can fail over to the passive node. If SERVER2 goes down, you will no longer be able to update the index, but the users will still be able to run queries against the existing index; from the users' perspective Search is still available (see Figure 3-4).

![Diagram of a medium-sized server farm](image)

Figure 3-4: A medium-sized server farm

Note that in this setup SERVER1 and SERVER3 are behind a network load balancer. This is only for the WFE portion, to balance the user load. It is not a requirement of the query service running on the boxes. Also, in this setup, we defined SERVER2 as WFE and INDEX, although that is not a requirement. This is done purely for performance reasons, as outlined in the previous section.

**Larger Farms**

Now that you have achieved HA, it is time to look at ways to achieve more scale (see Figure 3-5).
SERVER1 and SERVER3 are dedicated solely to providing web pages to the users. SERVER2 is still performing the index role, but now, instead of pointing to itself for indexing the SharePoint content, it is pointing to SERVER6. The reason for this change is performance, once again. Because you can only scale SERVER2 up in some scenarios, it is necessary to have a dedicated web target, because SERVER2 can’t handle the additional load of rendering the pages. SERVER6 lives only to serve pages for indexing, and it is purposely left out of the NLB rotation. SERVER4 and SERVER5 are dedicated exclusively to responding to search requests. You could continue to add these dedicated servers to achieve whatever level of scale you need. Keep in mind that this is not a recommended farm, just a list of talking points; however, there is no reason that you couldn’t run this setup if it met your needs.

For looking at sizing your SharePoint farms, Microsoft has released System Center Capacity Planner 2007, which includes templates for WSS v3 and Microsoft Office SharePoint Server 2007, at www.microsoft.com/systemcenter/sccp/default.mspx. While it does not specifically cover dedicated search scenarios, it can be used to help size your overall farm, including Search.

**Search Software Boundaries**

Several of the boundaries in Table 3-1 are managed through admin tasks and are covered in the next chapter.
Chapter 3: Planning and Deploying an Enterprise Search Solution

Table 3-1

<table>
<thead>
<tr>
<th>Item</th>
<th>Recommended Limit (* for hard limit)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexes</td>
<td>One per SSP*</td>
<td>Three SSPs per farm recommended, 20 SSPs hard limit.</td>
</tr>
<tr>
<td>Servers per index</td>
<td>One*</td>
<td>Keep in mind that you cannot spread an index across multiple servers, and you cannot aggregate multiple indexes into one result set.</td>
</tr>
<tr>
<td>Items in index</td>
<td>50 million</td>
<td>This is a tested maximum.</td>
</tr>
<tr>
<td>Content sources</td>
<td>500 per SSP*</td>
<td></td>
</tr>
<tr>
<td>Start addresses per content source</td>
<td>500 per content source*</td>
<td></td>
</tr>
<tr>
<td>Scopes</td>
<td>200 per site</td>
<td>While this is a functional limit, it is not very usable from a user perspective.</td>
</tr>
<tr>
<td>Rules per scope</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Display groups</td>
<td>25 per site</td>
<td>Used to display scopes.</td>
</tr>
<tr>
<td>Crawl rules</td>
<td>10,000 per SSP</td>
<td></td>
</tr>
<tr>
<td>Keywords</td>
<td>15,000 per site</td>
<td>Five synonyms per keyword.</td>
</tr>
<tr>
<td>Best bets</td>
<td>10 per keyword</td>
<td></td>
</tr>
<tr>
<td>Crawled properties</td>
<td>500,000 per SSP</td>
<td></td>
</tr>
<tr>
<td>Managed properties</td>
<td>100,000 per SSP</td>
<td>100 crawled properties per managed property.</td>
</tr>
<tr>
<td>Authoritative pages</td>
<td>200 per level of relevance</td>
<td>Four levels available in total.</td>
</tr>
<tr>
<td>Results removal</td>
<td>100 at a time</td>
<td></td>
</tr>
<tr>
<td>Crawl log entries</td>
<td>50,000,000 per log</td>
<td></td>
</tr>
</tbody>
</table>

Hardware Sizing Considerations

When buying hardware for your farm, Search is just one of the many considerations that go into your purchase. That makes it quite difficult to discuss exactly what you should purchase. Instead, we will review the different factors that introduce greater requirements.

Before discussing the various components, and considerations for each of them, it is important to discuss the network. While Search and indexing may be capable of running efficiently on a 100-Mbps network, Gigabit capacity is still highly recommended. Gigabit allows for more efficient SharePoint backups. Why is this important to Search? Because the index process is paused while backups are being run.
Chapter 3: Planning and Deploying an Enterprise Search Solution

For all servers in your farm, a 64-bit architecture is recommended over a 32-bit one. Microsoft has more information on this recommendation at http://technet.microsoft.com/en-us/library/cc261700.aspx. This is the last version of SharePoint that will be available on the 32-bit platform. While the upgrade scenarios have not been announced for the next version, it is probably safe to assume that 64 to 64 will be less complicated than 32 to 64.

**The Index Server**

As you weigh the following factors, it is safe to assume that more hardware is better than less. Since the index cannot be spread over multiple servers in the farm, the only option you will have for better index performance will be adding hardware to the box. Also, because this server could be considered a single point of failure, we recommend that you use fault-tolerant hardware.

You should buy at least two CPU cores and add capacity to meet your needs as you look into things. The number of index threads used will be based on the number of cores you have in the machine and the performance setting you specified when starting the Office SharePoint Server Search service. In Table 3-2 each processor core is counted individually when using dual or quad core processors. Hyperthreaded CPUs count as 1.

<table>
<thead>
<tr>
<th>Indexer Performance</th>
<th>Total # of threads</th>
<th>Maximum # of threads per host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced</td>
<td># of CPUs</td>
<td># of CPUs</td>
</tr>
<tr>
<td>Partially Reduced</td>
<td>4 times # of CPUs</td>
<td># of CPUs plus 4</td>
</tr>
<tr>
<td>Maximum</td>
<td>16 times # of CPUs</td>
<td># of CPUs plus 4</td>
</tr>
</tbody>
</table>

Because of the maximum of 64 threads, there does reach a point where additional cores will not net further performance increases. You could possibly take advantage of the additional CPUs by utilizing additional SSPs, but, since each one is unique, there is still no increase in the performance of the individual SSP’s crawl. You might see greater performance gains from investing in faster clock speed with four cores than lesser clock speed with eight cores.

Speaking of SSPs, there is another important factor in determining the number of CPUs you should have in your servers. While one SSP is always ideal, this doesn’t always fit the business requirements. If you plan to host additional SSPs, plan to add more CPU cores.

The types of content you are crawling can also affect CPU needs. When indexing content that is not plain text, the indexer must use an IFilter (for more information please check the section on IFilters in Chapter 4) to read the text. Various IFilters have different CPU overheads, depending on whether they are multithreaded. Check into the different IFilters you will need in your environment and their capabilities, when determining the number of CPUs you will need. The IFilter team now maintains a very helpful blog at http://blogs.msdn.com/ifilter.
Chapter 3: Planning and Deploying an Enterprise Search Solution

The location of the content to be indexed can be a big factor also. Indexing SharePoint 2007 sites and Exchange public folders is a very efficient process. Both of these sources have the ability to use a change log that tells the indexer what files are new or have changed; that way only the updated files are downloaded. File shares and traditional HTTP sources are not that good, but still not bad. With these content sources, the index process must make a round trip to each document, but it can determine whether the file has changed without downloading the entire document. Then, you have SharePoint v2 sites. Unfortunately, the index has to crawl the entire site and download each document to check whether it has changed. If you plan to index these sites, plan for the added overhead.

The language of content also can increase your CPU needs. Asian non-white-space languages (such as Japanese) and German have very resource-intensive word breakers. If you will be crawling a large amount of these documents, you should plan for the maximum CPU power you can get.

Microsoft testing, provided on TechNet, in its lab, found the net percent gain in crawl speed as one added CPUs; the results are shown in Table 3-3.

<table>
<thead>
<tr>
<th># of CPUs</th>
<th>% improvement in crawl speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10.89</td>
</tr>
<tr>
<td>4</td>
<td>19.77</td>
</tr>
<tr>
<td>8</td>
<td>30.77</td>
</tr>
</tbody>
</table>

As far as RAM goes, there are no steadfast rules other than *more is better*. Typically, you would not want to have less than 4 GB of RAM. Four GBs should work well, up to about a million documents. All documents to be indexed are buffered in memory, and then processed in memory before being written to disk. It stands to reason that as you add more CPU cores to the index server, you will have more parallel threads running, trying to buffer more documents in memory. So, if you buy eight CPUs, you will be able to take advantage of more RAM than if you buy a server with four CPUs. Keep in mind that there is a maximum number of threads that can be allocated per host (see Table 3-2), so having more CPUs isn’t always going to help.

Other considerations in regard to the amount of RAM on your index server are the other roles the server will be performing. If, in the name of efficiency, you have chosen to make your index box also a WFE, then your need for RAM will increase, because the worker processes will have to spin up for each web application that is being crawled. This can greatly increase the amount of required RAM. The amount of RAM on this server is not as crucial as on the query server. If the index process needs to page while indexing, that is OK. Hopefully, if this is the case, the system page file is on dedicated disk to avoid IO impact.

Next, is determining the amount of disk space you will require on the index server. This is another place where you are going to make some bold, educated guesses. In the real world, it isn’t uncommon to see the size of the index range from 1 to 15% of the size of the content that you are crawling. In most environments, the size trends toward the bottom of that range, depending on your average document
size. The latest published guidelines from Microsoft are: If the average file size is 10 kb, your index will be 12%, but if your average file size jumps up to 100 kb, your index will be approximately 5%. Another thing to consider here is SharePoint content also typically produces smaller index sizes due to the nature of list items. Once you have determined your base index size, you still need to do a little more math.

To determine the amount of storage needed for storing and working with the index file on the server, multiply the size of the physical index file (from above) by 2.85. This is the amount of storage required. The reason that this storage is needed is that the master merge process allows one index to remain online while the new updated version is put into place. The size of your index will grow over time, meaning that your storage needs will grow by 2.85 times the amount of the index’s growth.

Also, when purchasing disks, it is very important to consider the disk performance of the storage drive. We recommend that you store the index on a RAID 1 or RAID 10 volume dedicated only to the index. This will provide you with optimal performance given the amount of disk I/O required. If you cannot put the index on its own physical disk, then create a logical disk and store the index there by itself. That way, unplanned growth of this file will not fill up the OS drive and bring the system to its knees. The Office SharePoint Server Search service, covered in Chapter 4, pinpoints the best place to store the index.

**Query Servers**

In most farms, the WFE and query roles are assigned to the same servers. If Microsoft Office SharePoint Server has taught us anything, it is that there is no such thing as too much CPU. At a minimum, you will need two cores. SharePoint is multithreaded and can take advantage of multicore processors very efficiently. The more user queries you need to support, the more cores you should use. User response time is also closely tied to actual CPU clock speed, assuming that there are no I/O bottlenecks. Keeping all of this in mind, there is a point where buying another query server (scaling out) is cheaper than building bigger hardware (scaling up). Of course, in keeping with the HA story, you should have at least two query servers in the farm to avoid downtime.

When possible, Search will cache user queries and respond from memory. The more RAM you can dedicate to this task, the fewer number of trips to the file system for the index, and the speedier the entire system. In a perfect world, you would have enough RAM to load the largest *.ci* file (located in the index folder) into memory. In fact, until you build your index, you will not know the maximum file size, and in indexes with more than 10 million items, the size becomes entirely too large to make this practical.

Your query server will have the same storage requirements as your index server. Also, you will still need to separate the index onto unique, fast disks.

**Database Servers**

In this section, we will not discuss database CPU or RAM requirements, but instead will concentrate on storage and disk optimization of the search property store database. The number of disk spindles is often the top performance factor on the database server. Ten or more is not uncommon in an enterprise deployment. The search property store database is created when you create an SSP and contains all metadata properties for the crawled content and the crawler history logs. The property store database is used in conjunction with the physical index for serving user queries.

Sizing this disk is the first challenge to overcome. Microsoft guidance says that the database will be four times the size of your index file. In fact, it depends. Stores with large quantities of metadata
(e.g., SharePoint) tend to have a much higher ratio than a file share with limited properties. We also said, however, that SharePoint content typically produces smaller index sizes. If the index were 1% of your content, and the property store ended up being 10 times the size of that index, you would still be only at 10% of your content for the store. See the testing section later in this chapter for more advice.

Just because you determined a planned size for your property store does not mean you are ready to starting buying disks. When optimizing SQL Server database for SharePoint there are several best practices you should follow. The first is TempDB. It should reside on its own, fast physical disk, preferably configured in RAID 10. TempDB is a database used by SQL Server for temporary transactions and intermediate queries, among many other things. SharePoint puts a large burden on this database, so this database has the highest IO requirements of any of the databases SharePoint uses.

The second database you should optimize is the search property store database. Slow IO on this disk is often the indexing bottleneck and can also cause high query latency (meaning that your users will have to wait longer to get search results). These factors mean that it’s preferable for this database to live on fast, dedicated disks (again RAID 10, if possible). Separating the data files and transaction log files is also highly recommended.

The property store database has two distinct roles serving user queries and storing crawl information. Therefore, it is possible to separate the database into two file groups to optimize performance even further. The advantage is that while a crawl is taking place, that IO traffic can be segregated from the IO traffic from user queries. There is a noticeable increase in user query times while an index operation is running. There is no official guidance available at this time on separating the database into two file groups, and it should not be attempted on your own. Testing is currently underway and a white paper should be available on the topic by the time you are reading this paragraph. Check out the Enterprise Search Team blog at http://blogs.msdn.com/ifilter for the latest information on this topic.

For additional information on optimizing the other SharePoint databases and their storage needs check out the white paper at http://technet2.microsoft.com/Office/f/?en-us/library/ca472046-7d4a-4f17-92b1-c88a743a5e3c1033.mspx. Once you have deployed Search, maintaining these databases is very important. While the SQL considerations are outside the scope of this chapter, you should take a look at the following paper, “SharePoint Database Maintenance,” by Bill Baer at http://go.microsoft.com/fwlink/?LinkId=111531&clcid=0x409.

Testing

Unfortunately, after reading all of the preceding information, you may now have as many questions as answers, especially when it comes to disk sizing. That is the nature of the beast when sizing hardware requirements. There is almost an infinite number of variables that go into the equation, and truly every environment is unique.

Ideally, to help in your disk purchase decision, you should do a Search proof of concept (POC) before purchasing your production hardware. This will give you the opportunity to crawl a portion of your data and get some baseline numbers. You could look at your index size versus the size and number of documents you index to get your own index size ratio. Then, you could compare your index size to the property store database size to get your own multiplier. Remember that capacity planning is nothing more than an educated guess about your hardware needs.
Performance Monitoring

Performance monitoring has been in the administrator’s tool bag for years, and fortunately the developers of Search have chosen to provide counters we can use to monitor the process. While there are over 100 different counters for monitoring, there are a few to consider from a crawl/hardware standpoint. These are the ones we will concentrate on.

The first step is to launch performance monitor from Start ➔ Administrative Tools ➔ Performance on the index server. Then, to avoid clutter, remove the three default counters by pressing the Delete key three times. This should give you a screen like the one shown in Figure 3-6.

![Figure 3-6: Performance monitor with no counters](image)

The first counters to add are Total docs, in first queue, and Total docs, in second queue, from Office Server Search Archival Plugin.

1. Click the + (plus symbol) on the menu bar.
2. From the Performance object drop-down, select Office Server Search Archival Plugin.
3. In the Select counters from list section, choose Total docs in first queue.
4. In the Select instances from list section, choose Portal_Content.
5. Click Add.
6. Repeat steps 3–5 for Total docs in second queue.
7. Click Close.
8. Start a full crawl of your content source and monitor the output.

If you have more than one SSP, then you will have multiple instances in the format of Portal_Content#1, and so on. If this is the case, you will need to use trial and error to find which instance is for the SSP you are currently trying to test. Once you figure out this piece of information, it will be consistent in the future.

Normal behavior for these counters is for one counter to increase steadily up to a maximum of 500 before returning to zero. The other counter will start to build up to 500, while the first counter remains at zero. Look at Figure 3-7 for an example of a server on the verge of a bottleneck.
When watching the results, pay more attention as the crawl gets closer to finishing. This is when issues are most likely to show up. The key thing to watch out for here is whether both counters show 500 at the same time. If this is the case, then you likely have a SQL IO issue. Try to optimize the TempDB and search property store databases.

Assuming that you have no problems with the archival plugin, the next thing to look at is whether you are getting into a starved state during the crawl process. A starved state implies that your system requires more resources than are currently available. During this period, crawl performance is hurt by the constant contention for resources. If you find yourself in a starved state, you should reduce the load. By reviewing Table 3-2, you can determine the maximum number of threads that can be allocated to a specific host. This is based on the performance setting; also keep in mind that you cannot exceed 64 threads. Those threads can be in one of three states:

1. **Idle Threads** — These are threads that are sitting idle, waiting for work. They will eventually terminate if they sit idle for too long. If the number of idle threads is consistently greater than the number of threads per host (found in Table 3-2), then you should consider crawling an additional host with the current crawl. Conversely, if the number is consistently zero, then you are in a starved state and performance is being degraded. Consider reducing the number of hosts being crawled. A starved crawl is constantly battling for resources and thus is very inefficient.
Chapter 3: Planning and Deploying an Enterprise Search Solution

2. **Threads Accessing Network** — These are threads that are waiting for information requested from the network or are downloading and filtering the requested information. The best way to tell the status is to look at CPU and network activity. If this number is constantly high, you are either network bound or waiting for a hungry host. A hungry host is a host that is consuming more crawl resources than is typical. These hosts fall into one of four categories:

   a. **Slow host** — A host that does not have the available resources to respond to all of the index requests being sent to it at a given time. Consider using crawler impact rules (covered in Chapter 4) to lower the number of threads allocated to these hosts. This will not cause them to crawl any faster, but it will free up resources for other hosts.

   b. **Hosts that require extra incremental crawl overhead** — An example is SharePoint v2. Because all content has to be read and downloaded to determine whether there were any changes, this can be very expensive to crawl.

   c. **Hosts rich in metadata** — These are sources, such as the BDC and the people import, where each item has an extensive number of properties. These sources put a heavier burden on the property store database.

   d. **Flat hosts** — These are hosts that contain a single level with lots of content. An example is a file share with ten thousand documents in a single folder. All ten thousand documents will go into the queue, but there is a thread limit per host, and only that many threads will process the huge queue.

If at all possible, you should avoid crawling more than one hungry host at a time. Hungry hosts hurt overall performance and will slow down the crawl times for your non-hungry-hosts as well.

3. **Threads in Plugins** — These are threads that are processing the filtered documents after they are handled by the threads accessing the network. Through the various plugins, they are updating the index and the property store database. If you see threads consistently in this state, you may have SQL IO issues, as noted above. This can also be due to high SQL CPU load.

The above threads are listed under Office Server Search Gather, under Performance Objects, in Performance Monitor. Also, you should look at these threads over time for trends, as they fluctuate very frequently. When counting the number of threads, do not count Idle threads. For example, if you had a quad core CPU, and you were running the maximum for the performance setting, you would use Table 3-2 to determine that you could have a maximum of eight threads per host. When you look at the performance counter, the total number of threads in the thread accessing network and threads in plugins cannot exceed eight. The number of idle threads is considered. If you saw that you had 10 idle threads, then you would know you had the capacity to crawl another host at the same time.

There is another performance counter object called filtering threads. This is the sum of idle, accessing network, and in plugins threads. This counter is not generally useful.

### Search Backups

When it comes to backing up your SharePoint farm, there are several special considerations for Search. Unlike your web applications, you can’t just back up a database, use a SQL Server maintenance plan, and then recover from there. While you could backup the search property database, that would not recover the physical index files, and without those you would have to recrawl your content. You also
cannot just back up the index folder and restore the files. Instead, you will need to use the out-of-the-box backup tool from central admin, or a third-party tool that allows recovery of the index. The reason for the issue here is that the database and index files must have a consistent state when they are recovered. Out-of-the-box backup ensures this, and some of the vendors do also.

For most companies, using the GUI version of backup from central admin is impractical because it cannot be scheduled. Luckily, the stsadm tool can be. Using `Stsadm.exe –o backup –directory` will allow you to specify the settings of your backup in a single command line and output the results to a folder. You can then put this command line into a batch file and use the built-in Windows scheduler to run this backup nightly.

While you can choose to back up the entire farm with this method, you may want to capture only the index, property store, and SSP database. These are the items you will need to recover Search. This is also possible with stsadm. You will need to use the `–item` command as follows:

```
Stsadm.exe -o backup -directory \yourserver\share -backupmethod <full or incremental> -item YourSSPName
```

By specifying your SSP name, you are able to tell the backup only to capture the SSP. Of course, the SSP includes the property store and the physical index for which you were looking.

Specifying the options for the command demands a UNC path. The backup should be run locally to the server to increase throughput. When the backup completes, the files can be picked up and written to tape (or the appropriate backup device). Why do it in two steps? The reason is speed. The crawl process must be paused while the backup happens. In crawls with millions of items that take days to run, the less idle crawl time you have the better.

For more information on using the backup and recovery tools in SharePoint please check out the white paper at [http://go.microsoft.com/fwlink/?LinkId=102839&clcid=0x409](http://go.microsoft.com/fwlink/?LinkId=102839&clcid=0x409). While the paper claims to be for small to medium databases, it is just as valid for enterprise ones.

### Index Server Recovery Options

As noted earlier in the chapter, the index server is not redundant. Fortunately, because the index is propagated to the query server(s), and the search property store database is stored on the SQL Server, if the index server is not available, this does not affect the users. It does prevent the index from being updated. For some environments, fresh search results are paramount, so they need options for recovering the index server as fast as possible. We'll describe some scenarios.

Start by creating your production farm, creating an SSP, and getting your index to a steady state. Steady state refers to when your index is completely up to date, so future incremental crawls will only be indexing new or changed content. In an enterprise, with millions of items in the index, this may take a full crawl and two to three incremental crawls. A full crawl is such a long-running operation (think days) that when the full crawl finishes, the index is already incomplete (missing the content that was added while the crawl was happening). The next incremental crawl may take a day or more to run (depending
Chapter 3: Planning and Deploying an Enterprise Search Solution

on the rate of change of your content) to index those items. Repeat the crawls until the index process has “caught up,” so that it is completely current; you then have achieved steady state.

Once you have achieved a steady state, you are ready to back up the SSP and its associated content. With this backup, you can now build a separate recovery farm. In the recovery farm, you can restore your SSP, and you now have two copies of Search available.

Keeping the two farms in sync is where you reach decision time. One option is to backup the SSP continually and restore to the recovery farm if it becomes necessary. This is the slowest recovery option available.

Another option would be to restore to the recovery farm every time you do a backup. This way your backup SSP is always ready to go.

The final option would be to have your recovery farm now do its own indexing. This would need to be scheduled to happen while the production crawl was idle to keep the two processes from fighting for resources on the target hosts. This option requires considerable investment in hardware for your recovery farm.

Failing over to the recovery farm would then require you to change your production web application to change their associations to this SSP running on the second farm. There is a lot of planning that will need to go into making this a successful transition. While our intentions are to recover Search, in reality we are recovering the entire SSP. So, things like profiles, the audience, the site setting, and the BDC should all be taken into account.

Using Federation to scale?

As you know at this point it is not possible to spread the same index across multiple index servers. The only option you have is to scale the index server up to achieve better performance. Interestingly, Search Federation (covered in Chapter 7) may give you another option.

If you had two large content sources that you wished to index, and you did not care as much about returning aggregated results as you did about just making sure that they both returned results, federation might work. Federation gives you your standard search results, and, on the page in a separate area, you also get search results from another source. That other source could be an Internet search engine such as Live.com, or it could be another SharePoint SSP. Interesting? In theory, in your farm you could have two index servers, each serving its own SSP. Then you could configure Search Federation to show search results from both locations. The results would not be aggregated and would not have their relevance combined, but both could be displayed on the same page.

This will not work in all environments, but it is worth considering if indexer scale starts to become an issue. Keep in mind that, if you are going to use this technique, you need to be very aware of the load you are placing on the SQL server.
Chapter 3: Planning and Deploying an Enterprise Search Solution

Conclusion

There are many decisions to be made as you plan your Enterprise Search deployment. Hardware purchases are difficult, but you are now equipped to make some educated decisions. Keep in mind, though, if this is going to be your production farm for everything SharePoint, and not just Search, you have a lot more planning to do. The time to take up a continuous cycle of monitoring and review is when you deploy your farm. Your content will grow over time, and with that the need to keep up a fine-tuned Search experience will follow.
Configuring and Administering Search

Before you can take advantage of the awesome power of SharePoint Search, you first need to turn all of the knobs and switches the right way to make it work. For some administrators, this will be as simple as enabling the server roles for indexing and querying, specifying the accounts under which the services run, creating a Shared Services Provider (SSP), and finally making sure that the administrators have scheduled the indexing of their content. Just a couple of simple steps and Search is up and running. While that is great, and it speaks highly of the product that you can make Search function work so easily, most administrators will need to look deeper. This chapter will cover all of the special features you can use to dial in Search behavior and performance.

Configuring Search from Central Administration

SharePoint Central Administration is where Administrators go to configure farmwide settings for both the application and the servers that run those applications. This is no different from Search. From Central Administration, you will configure the Search services and assign them to servers in your farm. You can also create settings that will help you to control the impact that Search has on both the SharePoint farm and the target content source that you are indexing. This is a very important consideration because very few things that happen within your SharePoint farm are as resource intensive as indexing content.

The Search Services

By starting and configuring the two services, Office SharePoint Server Search and Windows SharePoint Services Search, you are really defining the roles of the index server and the query server in your farm. The index role is responsible for the actual finding of various content sources you have specified, connecting to that data, breaking that data down, writing it to a physical index
Chapter 4: Configuring and Administering Search

file and a property store database that can later be searched against, and finally (if necessary) propagating that index to the query servers throughout the farm. The query role is not complex. This service will accept search requests from SharePoint users and will provide relevant search results to them based on the information in the index and the Search property database created by the index server.

The index and query roles are very important and are a big part of SharePoint's high availability and scaling story. They can both live on the same server in the smallest of configurations or can be spread across multiple servers in various large scenarios. These configurations have been covered extensively in Chapter 3.

For clarity, in this chapter, we will be using a three-server setup. The first server will be named WFE1 and will be set up to host the web front end (WFE) and query roles, this server will also host SharePoint Central Administration. The second server will be named INDEX and will host only the index role. The third server will be named LITWARESERVER. This server will not run SharePoint Server but will run SQL Server 2005 Standard with Service Pack 2. WFE1 and INDEX are running Microsoft Office SharePoint Server 2007 Enterprise Edition with Service Pack 1 and a post SP1 HotFix, KB947701, which can be downloaded from downloads.microsoft.com. The HotFix provides an update to Microsoft Office SharePoint Server, which integrated the features of Microsoft Search Server 2008 into Microsoft Office SharePoint Server. Some of the key features introduced by the update are the capabilities of Federated Search and a simplified administration dashboard.

The Office SharePoint Server Search Service

This service is the heart of SharePoint Search. You define the roles of the index and query roles in your farm by enabling this service in different capacities on different servers. The services can be found in Central Administration by clicking Services on Server on the Operations tab. If you are setting up your farm for the first time, you should see a screen similar to the one shown in Figure 4-1. The other required services in the farm have been preconfigured and are not covered in this book.

![Figure 4-1: Configure Services on Server](https://example.com/figure41.png)
In the center of Figure 4-1, you have the options for selecting different server roles. This is a very common point of confusion for many first-time administrators. Selecting the different radio buttons will change the services you see listed at the bottom of the screen. However, these settings have no effect on the actual configuration of the server. The choices are meant to be a guide, showing you which services you will need to start or configure to achieve certain farm layouts, and nothing more. Do not be alarmed if you notice that each time you return to Central Administration the settings have defaulted back to Web Server for medium server farms; your service configurations are still intact.

Creating a Query Server

From the figure, you can see that the server WFE1 is listed. This server will be configured as a query server. To do this, you will need to click on Start to the right of Office SharePoint Server Search. This takes you to the screen shown in Figure 4-2.

![Figure 4-2: Configure Office SharePoint Server Search Service Settings on WFE1](image)

There are two check boxes at the top of the screen that you can select. Each one will make additional configuration settings available to you. For WFE1, you would like the server to host the query role, so select the second check box, Use this server for serving search queries.

The first piece of information needed is Contact E-mail Address. This is the email address that SharePoint will leave in the logs of servers as SharePoint crawls remote content. This gives a remote server administrator the option of contacting someone, if for some reason the SharePoint index process is causing issues on their server. Typically for this email address, you would want to specify a distribution list email address, either a dedicated SharePoint Farm Administrators’ alias or a helpdesk email address.

For a Farm Search Service account, you will need to specify a domain user account and password. It is very important that this account be a unique, dedicated account not used anywhere else in the product. SharePoint will use this account as the default content access account and will grant it full read access to all SharePoint web applications that will be indexed. If you were to elevate this account so that it had administrator access to any of the SharePoint content, then you would be adding to your index inappropriate information, such as previous versions and unpublished items. This would mean that users’ search results would contain a lot of “junk” the users should not see.
Chapter 4: Configuring and Administering Search

This account will also be used as the login account of the corresponding Windows service that will be started on the server. Additionally, the account entered here will be added to the local security group WSS_WPG and granted the Log on as a service local security policy.

Anytime that you are configuring SharePoint and specifying an account for anything in Central Administration or Configuration Wizard, you should specify it in the form of domain\username. There are occasions when you will be able to provide only a username without the associated domain, and while it may seem appropriate to accept it, you will run into errors. From experience, these access denied errors are very difficult to track down because everything looks fine. Only after careful examination will you discover the missing domain.

Next, is Query Server Index File Location, which tells SharePoint where on the file system you will store the index files when they are propagated from the index server. This folder is then shared out using a standard Windows file share named searchindexpropogation, and the Farm Search service account (specified above) and the WSS_Admin_WPG local security group are given full control of the share. This allows the index server to update the copy of the index file continuously on the query server(s) as the index is updated by the crawl process.

The default location for the index is c:\program files\Microsoft Office SharePoint Servers\12.0\Data\Office Server\Applications. As discussed in Chapter 3, this file can grow to be quite large, and typically you would not want to store this on your C: drive. To change the location, simply specify the new path in the box provided.

SharePoint needs you to specify a username/password of a local administrator account that has access to share out the location. This will be a onetime use of the specified credentials. The account under which you ran the SharePoint install on the server required local administrator’s rights to the server and would be ideal to use in this case. You can also manually set up this share using stsadm.exe, the command line tool for SharePoint.

Now that you have provided all of the information, you can click the Start button at the bottom of the page. This will take you back to the Services on Server screen. The Office SharePoint Server Search service should be listed as started. The next step is to set up INDEX to be the index server.

Creating an Index Server

Toward the top of the screen, where it says Server: WFE1, click the down arrow and choose Change Server. This will provide you with a listing of all servers in your farm. Click the INDEX server, and you will be taken back to the Services on Server screen. Notice that only the Windows SharePoint Services Web Application service is started. This means that currently your server is playing the role of a WFE server. The pros and cons of having your index server also serve the role of the WFE are discussed in Chapter 3. For this example, we will use the WFE role on the index server.

To the right of Office SharePoint Server Search, click the Start button. You will be taken to a familiar screen with two check boxes. This time you should select Use this server for indexing content. Notice that Contact E-mail Address and Farm Search Service Account are prepopulated for you. You do not need to specify the password again. Also, you will find that the Index Server Default File Location is set to the C: drive and is grayed out. This is not a problem. When you create a new SSP later, you can specify the index storage location and choose a more practical location.
For the Indexer Performance settings, accept the default setting of Partly reduced. The setting primarily affects the relative load placed on the INDEX server and the database server hosting the Search database. Look at Chapter 3 for details on optimizing this setting.

Web Front End And Crawling is a very important setting in your multiserver farms, but it can be a little tricky. First, we’ll do a basic run-through of how SharePoint will crawl web applications hosted on the farm. Surprising to some is the fact that the indexer does not connect directly to the database and start streaming content. The indexer will look at the start address (URL) specified and will try to connect to that site. It will call DNS and ask for the address information, which will resolve either to your network load balancer or directly to one of your WFE servers. The indexer will then make the connection to the server and start pulling in content to index. In our simple environment, that would go something like this: INDEX begins to crawl http://portal, which is hosted within our farm. INDEX looks to DNS for information on which server hosts http://portal and is given the information for WFE1. INDEX then requests http://portal from WFE1. WFE1 has to query the LITWARESERVER (database server) for the actual content to render the page. Once it has all of the information, it responds to INDEX’s request. Now INDEX can break down the file and write the information to the local index file and to the search property database housed on LITWARESERVER. As updates are made to the local index file, they are propagated to the query server, which is on WFE1. That is a whole lot of traffic, and anything that helps reduce it can make a difference in performance.

A very common recommendation is to add the WFE role to the index server, and then set the index server as the dedicated crawl target. This will greatly reduce the amount of network traffic and will help you avoid overwhelming the WFEs that are serving user requests. Even though you are specifying the index server as a WFE, you would not include it in any network load-balancing rotations. The goal is to not have any user requests hit the server; instead, the server should only serve pages for the index process. Keep in mind that, because this server is now a WFE and has all web applications in the farm created on it, you will need to maintain this server differently from others. If you were to update the web.config files manually or install features (for example) on the WFEs in your farm, you would need to repeat those steps on the index server now, so that all content can be rendered correctly. Another caveat to this method is that you cannot mix system architecture across roles in your farm. This means that all WFE servers in your farm need to be 32 bit or 64 bit. Having a mixture is unsupported by Microsoft; the two types are incompatible. Now that your index server is a WFE, it will have to have the same architecture. There are two distinct ways to achieve this setup.

From the Search Service Setup screen (see Figure 4-3), you can choose to use a dedicated web front end for crawling and specify the server on the associated drop-down. Doing this will cause SharePoint to make a host file entry for all associated web applications. Proceed cautiously here. If you have multiple network cards or IP addresses associated with your server, SharePoint will use the first IP address it finds for the server and associate it with all web applications. If that first IP address isn’t set up to return the web application because you are using IP-bound web applications or because it is a dedicated IP for other tasks, the results can be crawl failures. If you try to update or remove the IP addresses manually in the host file, the changes you make will be automatically overwritten within a minute of your making the change by a SharePoint timer job.
Chapter 4: Configuring and Administering Search

A better solution for specifying a dedicated crawl target in all but the simplest of farms is to update the host file yourself. Leave the settings for Web Front End And Crawling as Use all web front end computers for crawling, and then go to the index server and manually update the host file located at `c:\windows\system32\drivers\etc\hosts` with a tool such as Notepad. Then you can control which IP address is used for each URL. A note of caution about this method. You should document these manual changes to the host file very well and be sure to communicate the changes to the entire SharePoint administration team. Host file changes can make it very difficult for server administrators to troubleshoot what seems to be peculiar behavior.

Depending on how you configured the accounts in your farm and the associated Windows permissions of those accounts, you may encounter another issue with choosing a dedicated crawl target through the GUI. If the account your WSS Timer Service is running as is not a local administrator of the index server, then you will get the error Event ID 6482 in your event viewer once a minute. The error message will be Access to the path ‘C:\windows\system32\drivers\etc\hosts’ is denied. You will also notice that the file has been deleted. This happens because the Timer account lacks permission to recreate the file after deleting it. For more details on correcting this issue, please check out http://blogs.msdn.com/jjameson/archive/2007/05/05/the-case-of-the-disappearing-hosts-file.aspx.

You have now successfully configured the Office SharePoint Server Search service. Before you move on to creating your first SSP, there is another service to discuss and some additional settings to see in Central Administration.

**Windows SharePoint Services Search**

This service provides indexing capabilities in farms where only Windows SharePoint Services v3 (WSS) is installed. So, why would it be useful in a farm with Microsoft Office SharePoint Server or Search Server 2008 installed? It has one purpose: to index the help system that is included with SharePoint. When you click the little blue icon at the top-right corner of the screen, you get the built-in help system, which includes the ability to search. The WSS Search service runs only to serve this simple purpose. Some farm administrators will choose not to enable this service in the farm at all. The downside to this, other than Help Search being unavailable, is that you will get a nagging error report that a required service is not running on your farm. Typically, the recommendation is to enable the service and set it to run once a day.

Sometimes this service will be listed as Windows SharePoint Services Search and at other times as Windows SharePoint Services Help Search, but don’t worry. Regardless of the name, the service is the same.

Because this service is primarily an indexing function and does not require any type of high availability, typically it is enabled on the same server that was chosen for the Microsoft Office SharePoint Server index role (in the example, this would be the server named INDEX). To get started setting up the service,
make sure that the server name at the top is the correct server, and click Start to the right of Windows SharePoint Services Search. You will be taken to the screen shown in Figure 4-4.

![Figure 4-4: Configure Windows SharePoint Services Search Service Settings on server INDEX](image)

For the service account, you will want to specify a dedicated account that has been granted no special permissions. Once again, SharePoint will handle granting the account any necessary rights it may need. If your security needs require absolute minimal permissions and account isolation, you should plug into a dedicated, unique account for the service. The login account listed on the corresponding Windows Server service will be updated. More typically, when you are using this service only to index the help system and you have some tolerance for account reuse, you would specify the same search account that you used for the Office SharePoint Server Search service. This way, you have only one account to maintain for Search Services. The decision is yours. The account entered here will be added to the local security group WSS_WPG and granted the Log on as a service local security policy.
Chapter 4: Configuring and Administering Search

On TechNet there is an excellent article that discusses in great detail the various accounts you will need to specify when configuring SharePoint, and the various rights they will require. If you are still having issues making decisions about accounts, please review the article at http://technet.microsoft.com/en-us/library/cc263445.aspx.

The Content Access account will be used to read the content that is indexed by the service. The account will be given full read access to the necessary SharePoint data. Once again, depending on your security account policies, you can reuse the same account as the service, or you can specify another, unique account.

Search database allows you to specify both the Database Server and the Database Name of the required Search database. Typically, the defaults are used here. Windows authentication is the default for the database connection and is highly recommended.

Indexing Schedule allows you to control when the service runs. Unlike the Office Search service, which has powerful and flexible search scheduling controls, the WSS Search service only allows you to specify when full crawls take place. There are no other scheduling options available. Depending on whether you are pre- or post- Service Pack 1 for SharePoint, the default settings for this service will vary. Typically, neither of the defaults, every five minutes (pre-SP1), nor hourly between random times (post-SP1), are appropriate. The choice of daily during a low usage time is ideal for this setting. Since the help system is not going to be updated, there is no reason to recrawl the content any more than necessary.

While it may be tempting to start the service once, let it create the index, and then stop the service (because updates are not necessary), this will not work. If you choose to stop the WSS Search service, the associated index will be deleted, and the search results from the help system will be unavailable. As an added bonus, if you manually stop the service from Central Administration, you can no longer use that database, and it must now be deleted manually from SQL Server.

After you have successfully created all of the preceding settings, it is time to click Start. You will notice a performance spike as the service spins up for the first time. This is due to the start of the initial indexing of content. The service is pulling lots of content out of the database for the first time, so the spike can be pretty high. It should finish after a couple of minutes. Typically, this service is started when building the farm for the first time, so the additional load should go unnoticed.

**Manage Search Service**

From Central Admin, there are still a handful of additional commands you can run to tweak the search behavior further. Interestingly enough, to “Manage Search Service” you don’t go back to the Service on the Server screen; instead, you need to drill into the Application Management tab. There, you will find a link labeled Manage Search Service under the heading Search. This screen (as shown in Figure 4.5) has three main sections: The first is Farm-Level Search Settings, where you configure farmwide settings for elements like proxy servers, timeouts, and impact rules. The second section is Query and Index Servers; it is only informational, providing you information and links to the various servers in the farm. The final section is Shared Service Providers with Search Enabled, and it is informational, also providing you with the status of the various SSPs and the status of the associated indexes of each one.
From the Farm-Level Search Settings section of the page, you can drill down to make some changes that are going to affect the server and its behavior. When you click the link Farm-level search settings, you are taken to the screen shown in Figure 4-6.
Chapter 4: Configuring and Administering Search

You should recognize the contact E-mail Address setting, as it was specified when you started the Office SharePoint Server Search service. You could easily change it here if necessary.

For Proxy Server Settings, the default setting is Do not connect by using a proxy server, and for most environments this is the proper setting. If you are in an environment where a proxy server is required to reach any of the content sources specified at the SSP level, then you should fill in the blanks.

For an address, you can specify either the IP address of the proxy server or the NETBIOS name of the server. Under Port, specify the listening port of the proxy server. You can see the proxy server settings in Figure 4-7.
Chapter 4: Configuring and Administering Search

The option Use these proxy settings for access to federated sites is a new setting, referring to whether the specified server should be used for contacting federated search sources. If you check this box, then all federated sources will be contacted using this proxy server. If you would prefer to specify the proxy settings separately for each SSP that uses federated connections, then you can make these settings in the SSP’s web application’s `web.config` file. These settings will not override the check box above, and generally should only be used when absolutely necessary.

If you would like to use a proxy server for some content source connections, but not for others, then check the box Bypass proxy server for local (intranet) addresses. You can fill in the addresses to which the indexer should connect directly, bypassing the proxy server altogether.

Timeout Settings are for controlling how long the index process should wait while trying to contact content sources before discontinuing the attempt. The default setting is 60 seconds and works well for a default. You might need to increase this number if, for example, you are crawling slow or distant remote content sources, such as an old web server in another office somewhere across an ocean, or if you are working in a slow test/dev environment, such as a virtual PC running on your laptop. If you are seeing timeout errors in your search crawl logs, then this is the place to make changes to bump up the timeout settings. Be prudent with your changes, though. This timeout could adversely affect your normal crawl processes if for some reason a content source were unavailable.

SSL Certificate Warning Configuration is another common check box in testing scenarios. This check box will tell SharePoint, when it is attempting to index a secure content source (HTTPS), that it should ignore certificate errors that are typical of self-signed certificates. This is fine if you fully trust the content source and are expecting errors.

Once you are done making all of these changes, click OK to commit them. This will return you to the Manage Search Service screen.

Also available under Farm-Level Search Settings is Crawler impact rules. These rules control the load that the index process can put on a remote content source. These rules are often necessary when crawling external servers. It is very conceivable that a strong SharePoint index server could overstrain a remote server while it is being indexed, which in turn leads to unhappy administrators, who may block your future access to their servers. By setting crawler impact rules, you can lower the burden placed on those remote servers. Click the link to be taken to the rules screen.
When you first arrive, there will be no rules set up by default. You will need to click Add Rule to create the first rule. Here, you can specify the Site, and then the Request Frequency, as shown in Figure 4-8.

The figure shows a rule that says to crawl all *.com sites by requesting one document at a time. The available settings are shown in Table 4-1.

<table>
<thead>
<tr>
<th>Site Setting</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpecificName</td>
<td>Uses this rule for a specific name. Example: portal in this box would control the crawl settings when contacting <a href="http://portal">http://portal</a> or \portal.</td>
</tr>
<tr>
<td>*</td>
<td>Use this rule to affect all content sources.</td>
</tr>
<tr>
<td>*.com</td>
<td>Use this rule to affect all .com domains. Example: <a href="http://company.com">http://company.com</a> and <a href="http://www.anotherurl.com">http://www.anotherurl.com</a> would both be affected by this rule.</td>
</tr>
<tr>
<td>?</td>
<td>This allows you to have just one variable character. Example: server? would affect <a href="http://server1">http://server1</a> and <a href="http://server2">http://server2</a> exactly the same way.</td>
</tr>
</tbody>
</table>

After specifying the site, you then need to set the Request Frequency. You have two choices for this setting: Simultaneous requests or Time to wait. Simultaneous request will determine the number of documents that the index process will request at one time, and it can be set to 1, 2, 4, 8, 16, 32, or 64. Timeout will tell the server how long it should wait, in seconds, between requests. This can range from 1 to 1000 seconds.
Chapter 4: Configuring and Administering Search

As you start to add multiple crawl rules, it is important to keep in mind that these rules are evaluated in order, and once a rule that is relevant to the current content source is found that rule is applied, and the remainder of the rules are not evaluated. Take Figure 4-9 into consideration.

![Figure 4-9: Crawler Impact Rules in the incorrect order]

In this example, the rule *.microsoft.com will never be used. Why? Because when the content source, http://support.microsoft.com, is evaluated the rules are checked in order. Support.microsoft.com matches the rule *.com, so it will be indexed one document at a time. In order to change this and achieve the desired behavior of crawling at the rate of one document every 10 seconds, the rules will need to be reordered. To do this, click the drop-down to the right of *.microsoft.com and change it from 2 to 1. This will automatically refresh the screen and change the setting of *.com to 2. Now the rules will behave as expected (see Figure 4-10).

![Figure 4-10: Crawler Impact Rules ordered properly]

**Manage Content Database — Search Server 2008**

As you may have noticed at this point, when you are managing content databases, you can specify what search server the database will use. To check this out, open Central Administration, click Application Management, and in the SharePoint Web Application Management section, click Content databases. Now you can view all the databases for the current web application. Click on a database’s name to see its settings. In the middle of the screen, you should see Search Server, as shown in Figure 4-11.
Typically in your Microsoft Office SharePoint Server/Search Server 2008 setup, this should remain at its default setting of no server specified. The reason is that you will be associating your web applications with an SSP that will use the Office SharePoint Server Search service to index your content and will allow you to consume search results from there. Having the WSS Search service create a duplicate index that you will not use is less than ideal. The mostly likely situation in which you’d use this functionality with Microsoft Office SharePoint Server or Search Server 2008 is when you have created a web application that, for one reason or another, you have chosen not to index through your SSP. This may be due to the nature of the content or possibly the sheer volume of content. In this case, you can specify a search server for the content database, and then use WSS Search functionality instead of providing the Microsoft Office SharePoint Server/Search Server 2008 experience. This is not typical, but it is another great example of the flexibility that Search provides to accommodate the most random of scenarios.

Configuring Search from the Shared Services Provider

The Shared Services Provider (SSP), as it is known, is really the core of the search administrator’s experience. Most of the settings made in Central Administration were “set them up once and forget them” and were typically set during the initial farm configuration. The real fun is had through manipulating the SSP. The SSP is a set of centrally managed services that can be consumed by multiple web applications. Covering the SSP completely is outside the scope of this book. In this section, we will only be reviewing the details of how Search works on the SSP level. If you would like an overview of the SSP as a whole, please check out http://msmvps.com/blogs/shane/archive/2007/06/29/what-is-a-shared-service-provider.aspx.

Figure 4-11: Manage Content Database Settings
Creating or Editing the SSP Settings

In the sample farm, you have not created an SSP to this point, so that is the next step. We will skip over all decisions and steps that do not relate to Search, in the interest of brevity. To begin, from Central Administration, click on Shared Services Administration in the quick launch bar on the left side of the screen. From there you should choose New SSP.

On the new SSP screen, you will first need to make choices as to what to name the SSP and which web applications you will use to host the SSP and My Sites. In the example, you will use the default name of SharedServices1; you will use http://ssp to host the SSP and http://my to host my sites. Then, you will provide a service account and use the default SSP database. You come to the first search-relevant decision, the Search database.

The Search database, or property store, as it is also called, is used to store all relevant information about the crawled content. While the actual words inside the content are stored in the index file, which physically resides in the server’s file system, everything else lives in this database (such as associated metadata properties, file security information, and additional information that helps in calculating the relevancy of the content). Additionally, crawl and history logs are kept here. As you can see, this is a very important database, and it is required to be a top performer for an optimal search experience. In the example farm, you will use the default value for the name.

Now you can specify the index server and the location of the physical index on the index server. In the example farm, the index server is INDEX and will be grayed out. It is grayed out because there is only one index server in the farm, so providing a choice makes no sense. For the path for index file location in the example farm, you are going to specify a custom location of \MOSS Index, and then click OK. After watching the pretty green gear spinning for a minute, you will have an SSP ready for configuring Search. The index folder will automatically be created for you with appropriate security assigned.

If you ever need to view or edit these settings, you can come back to this screen by clicking Shared Services Administration from within Central Admin. Once there, you can hover over the SSP name, click the drop-down that appears, and chose Edit Properties. This will take you to a screen with all of your settings. If you choose to change the index location, SharePoint will delete your current index and then create a new index at the new specified location. This could be less than desirable; a better idea would be to use the command:

   stsadm.exe -o editssp -title YourSSPNameHere -indexlocation d:\new folder.

This will move your index from one location to the other. Once the move has been completed, you will need to delete the old index manually.

SSP Search Administration

Once you arrive at the home page of the SSP, under Search, click on Search administration. Here, you will load a Search dashboard similar to the one in Figure 4-12.
You should notice that everything that you need to use to configure Search or get a summary of what is currently happening is exposed through this interface. At first glance, it may seem a little overwhelming, but after a little practice you will come to appreciate the convenience that the dashboard provides.

As you probably have noticed, the Items in index is zero the first time you come here, and it is going to stay that way until you set up a crawl schedule. Crawl schedules are used to tell the server when it should index content, and what type of index operation it should perform. Each content source has its own schedule, so start with content sources, and see whether you can populate that index.

**The Default Content Source**

On the left side of the screen under Crawling, click the link for Content sources. You are taken to the Manage Content Sources screen and you should see the default content source Local Office SharePoint Server sites. Hover over the name, click the drop-down arrow, and click Edit. Now you are presented with the name and the details for the content source as shown in Figure 4-13.
Unfortunately, you have not done a crawl yet, so most of the information is marked as N/A. Reading through the headings, you can still see the power of the information that will be exposed. Information like last crawl duration can help you to understand better how long your indexing window really is, allowing you to tune your crawl schedule appropriately. It is also easy to compare your current, 7-, and 30-day averages. If your numbers seem to fluctuate randomly, it is a good indication that further investigation is necessary. Of course, if a quick peek shows that things are running as expected, that is a good time to get in a round of golf.

Continuing to scroll down the page, you will see additional information about this content source, as shown in Figure 4-14.

![Edit Content Source](image_url)

**Figure 4-13: Content source name and details section**

**Figure 4-14: Additional content source settings**
Chapter 4: Configuring and Administering Search

The first item you encounter is Start Addresses. For this default content source, SharePoint has automatically added an entry in the list for each web application associated with the SSP, saving you from the burden of updating this list each time you create a new web application.

For Crawl Settings you can specify whether you would like the crawl to spider to all sites within the web application, or to crawl only the site collection directly specified in the start address section. Typically, you would accept the default setting of Crawl Everything Under the Hostname for each start address because you would want the entire web application indexed.

Finally, you have discovered why the index is still at zero; by default, there is no crawl schedule specified. You have the ability to specify when a full crawl should happen separately from when an incremental crawl should happen. Before you look at the schedule setup process, we will discuss the difference between full and incremental crawls.

**Full versus Incremental Crawls**

In a full crawl, the index server will communicate with one of the WFE servers and will call the web service `sitedata.asmx`. This web service will return all URLs from the content databases and the metadata. The indexer will then issue GETs for all of the URLs and will begin to index all of the content in the web application. The full crawl then builds a new copy of the index, which it will propagate in the appropriate place, marking it as live at completion. When the new index is brought online, the old index will then be deleted.

Here are some of the reasons why a full crawl may be necessary:

- Software updates have been installed on the server. This could be a HotFix or service pack for SharePoint or the addition/removal of an IFilter.
- Crawl rules have been changed.
- A new content access account has been specified.
- The list of included file types has been modified.
- New managed properties have been added.
- To get updates to ASPX pages within SharePoint.

A full update will be automatically run in the following scenarios:

- A previous crawl was stopped.
- A content database was restored.
- This is the first time that a content source is being crawled.
- A corrupted index has been detected.

Incremental crawls of SharePoint are far less intensive thanks to the change log. The index server will communicate with one of the WFE servers and will call the web service `sitedata.asmx`. This web service will check the change log and will return all URLs of added, deleted, or changed content and metadata. The indexer will then issue GETs for all of the returned URLs and will begin to index the updated content. Instead of building a new index, it will write its changes directly to the current index.
That was the fancy version. Now, try the simple version. Full crawls will touch every piece of content you have in the given content source and index that content. This is a very resource-intensive process and can take a long time if you have lots of content. Typically, it is not practical to do a full crawl during normal usage, as the demands on the infrastructure can cause a farmwide slow down. They are best saved for off hours. Incremental updates will only index content that is different from the last time it was crawled. Thanks to advancements in the way that these updates are tracked, the incremental crawl process runs for a much shorter amount of time and for most environments can be run during normal usage periods. Incremental crawls during normal usage can actually reduce the length of time each one runs, reducing the hardware demand even more.

**Search Schedule**

Now that you have a better understanding of full vs. incremental crawls, we will explain how the scheduling process works. On the Edit Content Source screen, click Create schedule below Full Crawl. A window will pop up that looks like the one in Figure 4-15.

![Figure 4-15: Manage Schedules](image)

You will see that the first option is daily, weekly, or monthly. As you change any of the options, you will get new menus for configuring that option. For the example environment, choose daily. Now, choose how frequently you want the crawl to run. In the example, use Run Every 1 days. Next, you can specify a starting time. In an environment where the goal is to run a full index daily, you would run this update shortly after the nightly backup has finished. You want to be careful not to run your backups and full crawls at the same time. Both processes are very farm intensive, and running them at the same time will probably lead to random errors and corrupted data. Starting the crawl shortly after the backup is ideal, as the user load should still be very low, allowing the farm to be indexed as quickly as possible without causing user frustration from slow performance. In the example, set your starting time to 5:00 AM, and assume that your backup window is from 3 AM to 5 AM. You do not want the full update running again during the day, so do not select the box for Repeat with the day. Click OK to create the schedule.

Now, below Incremental Crawl, click Create schedule. For this example, choose Daily indexing running every 1 day. For a start time, choose to begin at 7:00 AM, so that the full update has time to complete successfully. You do want to repeat incremental crawls throughout the day; once an hour should be a good starting point. To set this, select the box Repeat within the day Every 60 minutes. Because you
want them to stop before the backups begin, choose For 1140 minutes. This way, the last incremental update will happen at 2:00 AM, long before the backups are started. Click OK to save the new schedule.

At this point, you could check the box Start full crawl of this content source, but you will look at a generally more convenient way to launch a full update on demand. For the moment, leave the box deselected and click OK.

**Additional Content Sources**

You will be taken back to the Manage Content Sources screen. We will now discuss adding additional content sources. The easiest way to do this is to click the New Content Source button and see what options you get, as shown in Figure 4-16.

The SharePoint Sites content source allows you to crawl any v2 or v3 SharePoint site. These sites can be hosted on the local farm or a remote farm. To crawl one of these sites, simply input the URL of the web application you wish to crawl in the Start Address box. Next, choose the option to either Crawl everything under the hostname for each start address (which will require you to enter only the root of the web application in the start address), or Crawl only the SharePoint site of each start address. In the latter case, you can specify any site address for the site address. If you choose a site collection or a subsite in the site collection as the start address, the entire site collection will be indexed.

The Web Sites content source allows you to crawl external non-SharePoint web sites, both HTTP and HTTPS are supported. Through the use of crawl rules, covered later in this chapter, you can even specify special login credentials for the sites. Once you specify a start address, you can then determine the crawl behavior for the site. Only Crawl Within the Server of Each Start Address will allow the crawler to follow all links encountered within a site to spider through the whole site, but it will not index any references to external links. Only Crawl the First Page of Each Start Address can be used when you wish to index only a specific page on a site. The final option, Custom – specify page depth and server hops, gives you complete flexibility. The first check box, Limit Page Depth, lets you determine how many layers deep the index process can go from the initial entry page. The second option, Limit Server Hops, is the dangerous setting. This setting tells the crawler the number of hops to other web sites it should index. If the setting is 1 and the crawler is pointed to http://www.msn.com, for example, you are telling the crawler that following external links to another web site (such as www.microsoft.com) and indexing that site is OK,
but following external references from that site is not allowed. In controlled situations, this is not dangerous, but what if you had said unlimited? You could find yourself trying to index the entire Internet. We doubt your hardware is quite ready for that; call it a hunch.

The File Shares content source allows you to index the content of file shares on your Windows Servers. You may specify the start address as `\server\share` or `file://server/share`. Once you input your start addresses, you have two choices for crawl settings. Choose the default of The Folder and All Subfolders of Each Start Address to index the start location and all subfolders. If you prefer to index only the specified folder, without the subfolders, then select The Folder of Each Start Address only. It is possible, through the use of crawl rules, to refine the behavior even further, excluding particular files or folders as necessary. This functionality will be discussed later in the chapter.

The Exchange Public Folders can allow you to connect public folders using the OWA URL. Like file shares, you have two choices for crawl settings. Choose the default of The Folder and All Subfolders of Each Start Address to index the start location and all subfolders. If you prefer to index only the specified folder, without the subfolders, then select The folder of each start address only.

The final content source listed in Figure 4-16 is Business Data. It should be noted that this content source is only available if you are using Microsoft Office SharePoint Server 2007 Enterprise Edition, as it is made available through the Business Data Catalog (BDC) feature. If you have added BDC applications to your SSP settings, then you can choose to index that data. There are two available options, crawl the entire BDC or crawl only specific BDC applications. There are many business decisions that should be considered before exposing your BDC through Search. These decisions are outside of the scope of this book.

This covers all of the preconfigured content sources available through Search. Each content source relies on an associated protocol handler to make the necessary connections. If you would like to index additional types of content, you will need to install a third-party handler or develop your own custom protocol handler. The most common example of this occurs when indexing Lotus Notes databases. There is a detailed article on TechNet that discusses this process. You can find the article at http://technet.microsoft.com/en-us/library/cc179339.aspx.

**Interacting with a Content Source**

There is one last thing to check out from the Manage Content Sources screen: the drop-down menu for each content source. Hover over Local Office SharePoint Server sites, and click the drop-down menu, as shown in Figure 4-17.

![Figure 4-17: The content source drop-down menu](http://technet.microsoft.com/en-us/library/cc179339.aspx)
Chapter 4: Configuring and Administering Search

This menu provides you with the most common operations you will need for a content source. It allows you to start a full or incremental crawl, view the crawl log just for the current source, pause or stop crawls, and finally, for all except the default content source, delete an unwanted content source. Since, to this point, you may not have run a full crawl, now would be an ideal time to do so. Select Start Full Crawl from the menu, and all of those reporting screens will finally show some meaningful data.

Crawl Rules

Crawl rules allow you to affect the crawl behavior of your content source directly through a very flexible rule system. While pointing Search at a content source and telling it to index away works for most scenarios, there are times you need the power to be granular. Perhaps you need to exclude a specific folder on a file share, provide a different username/password for a content source, or crawl a very specific portion of a web site. All of these options are available through manipulation of the crawl rules. To manage the crawl rules, click the link on the far left side of any of the Search Administration pages.

As with content sources, the easiest way to understand the potential is to click New Crawl Rule on the menu bar and try things out. Take a look at Figure 4-18 for the available options.

![Add Crawl Rule](image)

In Path you input the location that this rule should apply to. You can use wildcard characters, as necessary. If you want your rule to affect a file share use the syntax `file://server/share`; for a web site typically you would use `http://url` for your path. The onscreen examples work well to get you started.

For Crawl Configuration, you need to specify whether the rule is an Include or Exclude rule. For Exclude all items in this path you have the additional option of excluding complex URLs. A complex URL is one that contains query strings (aka ?) in the URL. Typically, you would use an exclusion rule if you wanted to exclude a specific path in a site, or as a catchall in complex situations. We will cover the complex situations after we finish covering the basics.
Include all items in this path is generally used for getting the crawl of your content source just right. Follow links on the URL without crawling the URL itself may be used when you have a table of contents (TOC) web page. You probably don’t want the TOC showing up in search results, yet you do want to index all of the links on the page. By default, Search ignores those complex URLs, but more than one client has used a custom solution when they needed to index those URLs as is; a simple rule using Crawl complex URLs took care of the problem. The option Crawl SharePoint content as HTTP pages is not very commonly used. The best example of when you might choose this option is in scenarios where you want just the content of the site indexed without the associated list and item permissions; this setting allows you to address that scenario.

Anytime you create an inclusion rule, you will get the choice of having a custom authentication setting. The ability to provide custom credentials is one of the most common reasons for creating an inclusion rule. For example, you cannot index a site using basic authentication without a crawl rule. We will walk through this scenario in a just a few more paragraphs. As you can probably guess, the first radio button Use the default content access account will use your default account. Choosing to specify a different content access account will provide you with new fields for entering the account username and password that you wish to use to authenticate; don’t forget to specify the domain if you need it. Specifying a different account gives you the option of allowing basic authentication, which SharePoint will not do out of the box. Since deselecting this box will cause your credentials to be transmitted in clear text, you should only do this for SSL-encrypted sites.

If your site requires a client certificate for authentication, then you can select the radio button Specify client certificate, which will then allow you to choose a client certificate from your index server.

The option Specify form credentials can be used to log in to the FBA sites. Simply input the URL of the login page for the site, and click Enter Credentials. This will open up a proxy window that will record your credentials so that Search can authenticate you against the form in the future. This will only work if the FBA site is using an input/submit type setup. It will not work if the site requires a multistep login process or uses some type of dynamic (Java, AJAX) method of posting back the page. Also, keep in mind that content crawled using this method will be returned for all users and will not be trimmed for security. If you need to security trim this content, a custom security trimmer will be required. See Chapter 8 for more information.

The final option is Use cookie for crawling, which allows Search to use a cookie for authentication. You can provide a URL and credentials to Get Cookie, or you can point to an existing authentication cookie on a computer on the network. Finally, you can specify the URLs of pages to which the server would be redirected if the cookie has expired, so that it can log the expiration. Also, keep in mind that content crawled using this method will be returned for all users and will not be trimmed for security. If you need to security trim this content, a custom security trimmer will be required. See Chapter 8 for more information.

Once you have configured your new rule, click OK at the bottom of the screen to save it. Once you have saved the rule, you will be taken back to the Manage Crawl Rules screen. Just like crawler impact rules (covered earlier in the chapter), these rules are evaluated sequentially, and once a rule applies, additional rules are not processed. This gives you a tremendous amount of power, but you must take great care to do things in the proper order. Try a couple of sample scenarios to see whether you can apply some of the power of the rules.
Crawling a Basic Authentication Site

For this example, you will add a content source to an SSL site hosted at https://wfe1. This site has been configured to use basic authentication.

1. In the Search Administration screen, click on Content sources.
2. Click New Content Source.
3. For Name, enter Secure WFE1 Site.
4. Content Source Type is Web Sites.
5. Start address is https://wfe1.
6. Crawl Settings is the default: Only crawl within the server of each start address.
7. Do not set up a schedule.
8. Click OK.

If you were to run a full crawl of the content source Secure WFE1 Site you would receive the error “The secure socket layer (SSL) certificate sent by the server was invalid and this item will not be crawled.” This results from the use of a self-signed SSL certificate that the index server has not been configured to trust. To deal with this error message, you can go into the Proxy and timeouts configuration and tell Search to ignore certificate errors.

1. On the left side of the page, click on Proxy and timeouts.
2. Scroll to the bottom of the page, and find SSL Certificate Warning Configuration.
3. Select the box Ignore SSL certificate name warnings.
4. Click OK.
5. Navigate back to the Search Administration Page in your SSP.

At this point, if you ran another full crawl, you would get the error message “Access is denied. Verify that either the Default Content Access Account has access to this repository, or add a crawl rule to crawl this repository. . . .” It seems that there are authentication issues. A quick check of permissions would confirm that the default content access account does have access, so that can’t be the problem. The problem, as it turns out, is that Search by Default will only use integrated authentication and will not pass its credentials to a site requesting basic authentication. We will need to create a crawl rule to do basic authentication.

1. From the left side of the page, click Crawl rules.
2. Click New Crawl Rule.
3. For Path enter https://wfe1/*.
4. For Crawl Configuration, select the radio button Include all items in this path.
5. At Specify Authentication select the radio button for Specify a different content access account.
6. For account enter credentials that have access in the form of domain\username.
Chapter 4: Configuring and Administering Search

7. Enter the password and confirm the password.
8. Deselect the box for Do not allow Basic Authentication.
9. Click OK.

Now would be the best time to try a full crawl of the content source, again. Assuming that you have not made any typos in accounts or rules at this point, your full crawl should be successful, and all of your content should be available to search.

Excluding Only a Portion of a Web Site

The basic structure of the example content source is https://wfe1, and for subsites https://wfe1/docs and https://wfe1/reports. The new requirements are not to index the https://wfe1/reports section of the site. To indicate this, you will need a new exclusion rule, but your rules will also need to be in the proper order to work correctly.

1. Open the Search Administration page.
2. Click Crawl rules on the left side of the screen.
3. Click New Crawl Rule.
4. For Path enter https://wfe1/reports/*.
5. For Crawl Configuration, select the radio button Exclude all items in this path.
6. Click OK. Figure 4-19 shows the current rule setup.

At this point, if you were to do a full crawl, what would happen? The contents of the reports folder would be indexed. Why? Because the rule https://wfe1/* would take effect, and the rule https://wfe1/reports/* would never be processed. You can use the box at the top of the screen; type in the URL you are trying to troubleshoot to see which rule it is using. This little interface makes troubleshooting easy. We need to reorder the rules to get the desired behavior.

1. To the right of https://wfe1/reports/*, click the 2 drop-down and select 1.
2. The screen will refresh and the rules will be in the proper order.
Chapter 4: Configuring and Administering Search

Hopefully these two scenarios will help you to understand better, when and how to use crawl rules. Just about anything is possible with a little content source and crawl rule creativity.

Crawl Logs

The crawl logs are a very handy tool for keeping track of exactly what Search is doing. These rich listings of everything that has happened, good or bad, during a crawl will help you to troubleshoot issues and to plan your crawl schedule better by reviewing start and stop times. By clicking on Crawl log, on the left side of the screen, you can get started exploring the logs, as shown in Figure 4-20.

![Crawl Log](image)

Figure 4-20: The crawl logs landing page, view by hostname

From the image, you can see that, by default, information is sorted by the hostname. At this point, you can click on the hostname to see all crawl log entries for the host, or you can click on one of the number of entries to see the results. For example, if you click on the 3 for successfully crawled items for wfe1, you will see the screen in Figure 4-21.

![Crawl Log](image)

Figure 4-21: Successfully crawled items in the log, viewed by URL

From the log, you can see all of the items that were successfully crawled and at what time. Also, at the top of the screen, you can see the highly flexible interface that is available for searching through the logs. The filtering options make determining the status of specific items in your index very simple.
Chapter 4: Configuring and Administering Search

Returning to the Search Administration home page shows you a dashboard-like view of the crawl logs, as shown in Figure 4-22.

![Figure 4-22: Active and recently completed crawl information](image1)

While active crawls give you real-time status updates of the crawl process, the Recently completed crawls web part shows you the previous crawls. By default, you will see the last three completed crawls with the basic information you need to determine whether things are running smoothly. A nice, yet forgotten, feature is that these are both web parts that can be customized to show additional information or more rows of data at a time. In the title bar to the right of Recently Completed Crawls, click the drop-down arrow and select Modify Shared Web Part, as shown in Figure 4-23.

![Figure 4-23: Modifying the shared web part](image2)

Scroll to the bottom of the web part properties box and expand Miscellaneous. Here, you can change the number of displayed rows from 3, and you can choose what columns show up. The available options are:

- Show crawl type
- Show finish time
- Show duration
- Show success count
- Show warning count
- Show error count

Once you have made any changes, click OK to save your settings.
Chapter 4: Configuring and Administering Search

File Types
Out of the box, Search is only configured to index 38 different file extensions; any other file types the crawl process encounters are simply ignored. For the complete list, click File types on the left side of the Search Administration screen. Adding an additional file type to this list is as simple as clicking the New File Type button in the menu bar. Then enter the file extension without the period (.), and click OK. Removing a file extension only requires finding it in the list, then hovering over the extension, clicking the drop-down, and selecting delete.

There are a few considerations that go with adding file extensions. The first is that for most non-text-based file types you will need to install an IFilter to allow the crawler to open the file and pull the words from the file (IFilters are covered later in the chapter).

Next, most files do not have an icon image associated with them, so when these file types are returned in Search, they will be presented with the Windows unknown icon. Microsoft has released a KB article that covers the proper procedure for updating the images at http://support.microsoft.com/kb/832809.

The final consideration is making sure that, if the file type with which you are working is going to be stored in your SharePoint site, it is not in the blocked file type list. This list is maintained in Central Administration under Operations and blocked file types. Out of the box, SharePoint blocks 90 different file types. Make sure that the one you are adding support for isn’t on the list, if you need to store the files in SharePoint.

Reset All Crawled Content
You can use this option to delete your entire index, leaving you with zero search results until you run a full crawl. Typically, this is only done in cases where you are having search issues and need to reset things to zero and start over. When you choose this option from the left side of the Search Administration screen, you will be presented with a check box to Deactivate search alerts during reset. Users can subscribe to search-based alerts through the Search Center and automatically be notified when a specific query has been updated. Deactivating these updates will keep users from getting unnecessary emails while you are purging and rebuilding the index. You can then reenable the alerts from the Search Administration interface.

Search Alerts
Search alerts are notifications to which users can subscribe through the user Search experience. These alerts provide notice that a particular query’s results have changed. The alerts can be subscribed to daily or weekly. As an administrator, you have the ability to turn these alerts on or off for the entire SSP. To do so, view the Search Administration page. In the center of the page, you will see Search alerts status and the current status. You may toggle that status by clicking the disable/enable link to the right of the current status. During index resets is a great example of when you may want to disable the alerts, temporarily.

Authoritative Pages
One of the determining factors that Search uses when calculating overall content relevance is the click distance from authoritative pages. Authoritative pages are administrator maintained and can be adjusted to help this weighting process at any time. By clicking the link on the left side of the Search Administration page, labeled Authoritative pages, you can review the options available. Please see Chapter 10 for more information.
Chapter 4: Configuring and Administering Search

**Federated Locations**

A unique feature of our Search Engine is the ability to do federated searches. This allows the users’ queries to be performed simultaneously against your index and other search tools or repositories, bringing all of the information together on the same page. This feature is so flexible and powerful that it has an entire chapter dedicated to it. Please see Chapter 7 for more information.

**Managed Properties**

Looking at your content sources, such as your SharePoint sites, you will see that they are full of rich metadata. This metadata provides a very rich experience, both when working directly with the associated items and when searching for those items. Not only does SharePoint Search give you the ability to search for those properties, but it also gives you a management interface for working with the properties to fine-tune the search experience. On the Search Administration home page, click Metadata properties in the leftmost column.

You first arrive at the managed properties view, but skip these properties for now, and start with crawled properties. There is a button on the menu bar that says Crawled Properties; click this button. We will start the discussion there.

Crawled properties is a list of every property that the indexer has encountered in any document while crawling. They are categorized for better discoverability by administrators. By clicking on a grouping, such as SharePoint, you can then see all of those properties. Now scroll down the page and click on *ows_LastName (Text)*, and you will see the gory details of the property. You will see the type of property it is; a listing of some sample documents; what, if any, managed properties it is linked to; and whether this property is included in the search index.

Managed properties are not discovered properties, but are instead manually generated properties that users can search against. These properties are made up of one or more crawled properties that you would like to combine as one. For example, *ows_lastname* and *ows_Last_x0020_name* are both crawled properties that contain last name information. So, there was a managed property created called **LastName** that is linked to both of the properties. Now, from Advanced Search in the Search Center, a user can search for **LastName**. When you enter the last name of Young, Search will check for documents with the property of either *ows_lastname* or *ows_Last_x0020_name* and will return any results for which the value is Young. Even though the two crawled properties are completely separate, you can manage them together to make the user’s search experience seamless.

The easiest way to find crawled properties is to use the search box above the menu bar. It will take your input and return all properties where the value is contained in the property name. If you searched for “last” you would receive *ows_lastname* and *ows_last_x0020_name* because last is contained in both names. This is easier than scrolling through hundreds of random properties.

Clicking on manage properties in the menu bar will return you to the listing of managed properties. As before, click the New Managed Property button. We will explain the various options as displayed in Figure 4-24 below.
Chapter 4: Configuring and Administering Search

![New Managed property screen image]

Property name is a required field and must be unique. Providing a description in the Description field is always a good idea, so the next person who has to manage these things has an idea of what you were trying to accomplish. The type of information is important because this will determine your options for searching the data in an advanced search later.

You are then given two choices: either Include values from all crawled properties mapped, or Include values from a single crawled property based on the order specified. The distinction is here for documents that contain more than one of the listed properties.

For example, imagine that you are looking at that LastName property that has both ows_lastlastname and ows_last_x0020_name mapped to it, in that order. You have also chosen the option Include values from a single crawled property based on the order specified. Now, you have documentA, which has both properties. Its value for ows_lastlastname is Smith, and the value for ows_last_x0020_name is Young. DocumentB only has the property ows_last_x0020_name with a value of Young. You then do a search for LastName:Young. DocumentA would not be returned, but documentB would be. Why? Your setting said to evaluate only a single property based on the specified order. For documentA, it only checked ows_lastlastname for the value of Young. DocumentB only had one property, which had a value of Young, so it was returned.

Add/remove mappings will allow you to maintain the list of associated crawled properties with this managed property. You may also directly add to a managed property while looking at a crawled property. Moving the properties up or down will only have an effect when you are using the Include values from a single crawled property, based on the Order Specified setting discussed in the previous paragraph.
Chapter 4: Configuring and Administering Search

The check box Allow the Property to be Used in Scopes determines whether scopes can be based on this property. Scopes will be covered later in the chapter.

**Shared Search Scopes**

Search scopes are created in order to allow users to perform more tightly focused searches. For example you have added content sources to your index for SharePoint Sites, File Shares, and Exchange Public folders. Typically, when one of your users goes to his or her site and performs a search, that person is using the All Sites scope. This scope will return a vast array of results from all of the content sources (assuming that the user has proper permissions to see the content) in the index. While, typically, this is what the user wants, for some queries that is too many results to dig through. In this case, the user knows the file that he or she is looking for is on the file share and would like to only get results from that location. By default, this is not possible, but with the creation of a custom search scope, this becomes possible. As an administrator, you can define custom scopes using a rule-based system. Then you will allow your users to perform searches across smaller sets of data returning more relevant results.

There are two types of search scopes that you can create. Shared search scopes are defined at the SSP level and are made available to all site collection administrators for use within their sites. These scopes will be the focus of this section. Additionally, as a site collection administrator, you may define local search scopes, which can only be used in a specific site collection. These scopes will be covered in Chapter 9.

To get started with scopes, click on Scopes from the left side of the Search Administration screen. From this screen, you can see both the shared scopes, and a listing of each site collection, and any local scopes they have created, as shown in Figure 4-25.

![Figure 4-25: Viewing all scopes for the SSP](image)

Clicking the New Scope menu button will allow you to create a new scope and will expose the various available rules and settings. The first information, Title and Description, should be pretty straightforward. For the Target Results Page, you can Use the default Search Results Page or specify a different page for searching this scope. This setting gives you the power to control the results page. Normally, when someone chooses a scope and performs a search he or she is sent to the default search results page, as expected. This option gives you the power to send them to a custom results page, instead. As you explore the customization options for search results pages (covered in Chapter 7), you can unlock the power. Once you have made your settings, click OK.
Chapter 4: Configuring and Administering Search

You are taken back to the View Scopes screen. To the right of your scope you will see Add rules. Click the link to start configuring the new scope. The add scope rule exposes all of the possibilities, as shown in Figure 4-26.

![Add Scope Rule](image)

Figure 4-26: Add Scope Rule

Select the radio button for Web Address to see the first set of options. Here, you can choose to include a specific folder, a hostname, or an entire domain. Don’t be misled by the name Web Address. You can use this type of rule for file shares or Exchange public folders just as easily.

The Property Query option will allow you to take advantage of the managed properties you defined earlier. Any of the properties you marked as Use in Scope can be specified here, allowing you to create content-specific scopes. Maybe you have a DocClass managed property in use. You could create an Invoices scopes and set it to return only documents where the managed property DocClass = invoice.

Content Source is a pretty simple, yet important, rule type. Use this rule to create a scope that returns all items from a specific content source. If you are indexing those file shares and want to provide your users with the power to query only those locations, this is your rule.

Finally, you have the All Content rule. This will return everything in the index. The default All Sites rule uses this rule in conjunction with an exclusion.

Regardless of what type of rule you choose, you can specify the behavior setting. The first choice is a standard Include. Essentially, any content that meets this particular rule will be included in the scope. For example, assume that you have two rules defined. One says content source = file share is included, and the other is set to DocClass = invoice is included. This scope would include all documents from the file share and all documents in the index with a DocClass of invoice.

The second choice is Require. Require means that all items in the scope must meet this rule. For example, assume that you have two rules defined, which are very similar to those above. One says content source = file share is included, and the other is set to DocClass = invoice is included. This scope would include all documents from the file share with a DocClass of invoice.

The third choice is Exclude. Exclude means that any documents that match the rule will be excluded from the index. For example, assume that you have two rules defined, which are very similar to those above. One says content source = file share is included, and the other is set to DocClass = invoice is excluded. This scope would include all documents from the file share that do not have a DocClass of invoice.
Chapter 4: Configuring and Administering Search

From the preceding three examples, hopefully you are starting to see some of the power available through the use of multiple rules and the modification of their behaviors.

Scopes are compiled on an automatic schedule, which is every 15 minutes. If you wish to disable this automatic schedule, or force the compilation to happen immediately, you need to go to the Search Settings screen (see Figure 4-27). To get there, return to the home page of the SSP, and, under Search, click Search Settings. Scroll to the middle of the page to the Scopes section, and click the Start update now link to force the scopes to update immediately. From this screen, you can also disable the automatic updates by clicking the link Automatically Scheduled and selecting On demand updates only.

The other options you have from the View Scopes screen are the abilities to copy a scope and to delete a scope. To perform either of these operations, hover over the scope name, click the drop-down arrow, and choose your option. Copying scopes can come in handy, especially when you want to recreate a site collection scope as a shared scope. A couple of quick clicks and you are ready to go.

![Figure 4-27: Scope configuration options on the Search Settings page](image)

Server Name Mappings

Server name mappings are used to return search results with different addresses from those in the index. For example, maybe you indexed your Exchange Public Folders using the path `http://exchange/public`, but because your users can access the Search results from the Internet, they need the results to be in the format `http://owa.company.com/public`. This can be provided through the creation of a server name mapping. To do this, click Server name mappings on the left side of Search Administration.

Once you are taken to Server Name Mappings, click New Mapping on the menu bar. Input the address that is listed in the index, and the address you would like to display in the search results, and click OK.

*For a properly configured SharePoint site, this setting should not be necessary. If your site is indexed via the address `http://portal` but is also available using `http://portal.company.com`, this should be addressed through alternate access mappings (AAM), and not through the use of server name mappings. AAM will make sure that the proper addresses are returned in search results based on the URL the users input into their browser. For more information, please check out the post at [http://blogs.msdn.com/sharepoint/archive/2007/03/06/what-every-sharepoint-administrator-needs-to-know-about-alternate-access-mappings-part-1.aspx](http://blogs.msdn.com/sharepoint/archive/2007/03/06/what-every-sharepoint-administrator-needs-to-know-about-alternate-access-mappings-part-1.aspx).*
Chapter 4: Configuring and Administering Search

Search Result Removal

Sometimes it is necessary to have specific items removed immediately from your index. Perhaps you found when you indexed the file share that one of the confidential folders you crawled had permissions set up incorrectly, exposing the data to everyone. You can’t risk waiting for the next full crawl to purge the results. In this case, Search result removal is your tool of choice.

From Search Administration, click Search result removal on the left side of the page. You are taken to an input box where you can enter URLs you want to remove. Two things will happen when you enter a URL and click Remove Now. First, the content will be removed from the index immediately; second, a crawl rule will be created to exclude that content from future crawls. This new crawl rule will be ordered first, so make sure that it does not conflict with any custom rules you have previously established.

If you no longer wish to exclude this content from search results, possibly after updating those permissions, then the only thing you need to do is go back to crawl rules and delete the rule, and then run a full crawl. There is no Search result removal list that is maintained. Every time you go into the setting you will be greeted with the same blank screen.

Search Reporting

Search comes configured from the start to do search logging. This logging provides two types of reports. Query reports reflect the quantity and location of searches requested for specific time periods. Results reports show you what the users are doing once they get search results back. Both are very powerful and useful in the hands of a skilled search administrator. All of these reports are exportable to either PDF or XLS format through the menu bar above each report. Just select a format, and click the Export button. The four query reports that are displayed as graphs or charts also can be changed to show their raw data by clicking the blue arrow on the menu bar to show the second page.

If you wish to disable Search Query logging for some reason, you can do so through the Search Administration interface. In the center of the screen, you will see Query logging along with its current status. The link to disable/enable this function is located to the right of the current status.

Queries Report

Out of the box, the system does a good job of tracking how your users are using Search. The five query reports are the first example of this. These reports are perfect for providing management with details on how Search is being used in the organization. Access to this report is available from the bottom left of the Search Administration page.

Queries over Previous 30 Days reflect the number of searches run, broken down by day for each of the last 30 days. Queries over Previous 12 Months is the same type of report, but compiled by month. Both of these reports give you just general usage numbers for how much people are using the search system.

Queries Per Scope over the Previous 30 Days will help you to see what search scopes are seeing the most traffic. Scopes that are seeing very little or no usage should be removed, as they are just consuming system resources and causing users to scroll past them to find the scopes they wish to use.
Top Queries over Previous 30 Days is an interesting report in that it shows you what your users are looking for the most. Sometimes the leaders in this report can be good indications of missing links in your site structure or taxonomy. If you see that a major site or piece of content is constantly being searched for, this is a good indication that users are having a hard time finding it. This could be due to the content being too many clicks deep from the home page or not being where your users expect it to be in the navigation. When you are looking to revise your site structure, this report can be helpful.

Results Report

If the query reports are primarily for delivering usage statistics to decision makers, then results reports are perfect for search administrators who want to fine-tune the search experience. These five reports provide insight into exactly what your users are doing with their queries. All of these reports are for the past 30 days.

Search Results Top Destination Pages is a report that will help you determine where users end up the most often after performing a search. As in the Top Queries report described earlier, these can sometimes be indications of pages that are hard to discover through navigation.

Queries with zero results can be quite worrisome. These are things your users have taken the time to search for, expecting to get results, and in turn they were greeted with a no results message. Most of the time these are entries with typos or items not expected to be found. Still, reviewing this report is important. Consider, for example, if you ran your query reports and found that the number one query performed over the past 30 days was “Company Picnic,” and when you ran the results report you saw the number one query with zero results was “Company Picnic.” Is this a problem? Say yes as loud as you can. While you may think the company picnic not being found in the portal is not a big deal, your users don’t see it that way. The users see it as, “This stupid Search Engine isn’t smart enough to find the company picnic page! I could never rely on it to find information critical to my job!” Now you have undermined the confidence in your entire deployment over a simple search query. Sounds extreme, but it has happened before.

Most Clicked Best Bets is really just a pat on the back report. This report is telling you what best bets are getting used the most. Hopefully, they are the ones you thought of and set up, so you can tell everyone how smart you are. Queries with zero best bets will help you make that next golden pick by reflecting those top queries that are yet to have a best bet created.

Hidden at the bottom of this page is arguably the most important report, Queries with Low Clickthrough. Clickthrough refers to how often a user clicks on one of the search results that were returned. Say that the user searches for invoices and gets results. The user then clicks on one of the results; this is a clickthrough. Results that are clicked on a very low percentage of the time tell you that something is wrong. A user came to page, ran a search request, got results back, but then chose not to click on any of them. This should be a red flag that you are not getting good results from the query. When you find queries with low percentages, they are prime candidates for keywords and best bets. You, of course, will need to go to the user community, determine what they expect to see when they run the query, and figure out why they aren’t seeing it.

The Other Search Settings

From the home page of the SSP you may have noticed that under Search there are two options that have not been covered up to this point: Search Settings and Search Usage Reports. The reason for their omission is that all functionality provided in these two settings is exposed through the Search Administration
Chapter 4: Configuring and Administering Search

interface, save one: the ability to start a scope update manually; but this feature was explained previously
in the scope section. Both of these menus are here mostly for legacy reasons, as they were the old way of
administering search in Microsoft Office SharePoint Server.

Configuring Search Locally on the Server

There are a handful of Search-related configurations or tasks that are required to be run locally from the
server. Installing third-party software, such as an IFilter, and digging into the server error logs are the
two main tasks. It is also possible to make many of the service- and SSP-level search settings through
the command line tool stsadm.exe, but this is often not practical, since the modifications can be made
through the browser without giving search administrators elevated permissions.

IFilters

IFilters are a core Microsoft search technology that is not unique to SharePoint. IFilters provide the ability
to read proprietary file types, so that the document can be indexed. The simple explanation? If you
have ever tried to up a DOC file using Notepad, you know that you got nothing but random garbage
characters, and you could not read the text. The same would be true for the Search crawler without
IFilters. Now, out of the box Search includes an IFilter for DOC files along with 48 other file types, so you
don’t have to do anything. (For a complete listing of included file types and IFilters please see
OneNote (.one) files? Then you would need to install an IFilter.

IFilters are unique in that there is no standard way to install them. Each IFilter is unique and should
come with its own set of instructions. Typically, when you are looking for an IFilter for a new file type,
the first place to check is with the product manufacturer. There are also third-party web sites that
provide or sell IFilters. Additionally, if you cannot find what you are looking for, IFilter is a technology,
so you can always roll your own. Finally, there is now a Microsoft blog dedicated to IFilters. You can find

Installing the Microsoft Filter Pack

Since we brought up the example of not being able to index OneNote files with SharePoint, we will dive
into how to install the Microsoft Filter Pack, which includes the ability to index .one files along with
several other types. To obtain the pack and for the official installation information go to http://
support.microsoft.com/?id=946336. Here we will walk through the install:

1. Download the appropriate build for your system, whether that is 32 bit or 64 bit, to your index
   server. IFilters only need to be installed on the index server.
2. Run the downloaded file.
3. Click Next at the welcome screen.
4. After reading the license terms, if you agree, click I accept, and then click Next.
5. Click OK at the success screen. Don’t be alarmed that the installation completes almost instantly.
6. Navigate back to your SSP.
Chapter 4: Configuring and Administering Search

7. Click on Search Administration.
8. From the left-hand column, click File types.
9. You will now need to add the file types by clicking New File Type on the menu bar.
10. Enter xlsb and click OK.
11. Repeat steps 9 and 10 for the file extension one and zip.
13. Navigate through the registry to HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office Server\12.0\Search\Setup\ContentIndexCommon\Filters\Extension\ (be careful navigating to this key, there are very similar keys, and it is easy to end up in the wrong place and not realize it).
14. Right-click on Extension and select New Key.
15. Name the key .xlsb (notice the . this time).
16. Open the key and edit the (Default) value.
17. Set the value to {312AB530-ECC9-496E-AE0E-C9E6C5392499} and click OK.
18. Repeat steps 14–17 for the following extensions. Note that the value you insert varies by key.

<table>
<thead>
<tr>
<th>Extension</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>.zip</td>
<td>{20E823C2-62F3-4638-96BD-90F4F6784EBC}</td>
</tr>
<tr>
<td>.One</td>
<td>{B8D12492-CE0F-40AD-83EA-099A03D493F1}</td>
</tr>
<tr>
<td>.Vsd</td>
<td>{FAEA5B46-761B-400E-B53E-E805A97A543E}</td>
</tr>
<tr>
<td>.Vss</td>
<td>{FAEA5B46-761B-400E-B53E-E805A97A543E}</td>
</tr>
<tr>
<td>.Vst</td>
<td>{FAEA5B46-761B-400E-B53E-E805A97A543E}</td>
</tr>
<tr>
<td>.Vdx</td>
<td>{FAEA5B46-761B-400E-B53E-E805A97A543E}</td>
</tr>
<tr>
<td>.Vsx</td>
<td>{FAEA5B46-761B-400E-B53E-E805A97A543E}</td>
</tr>
<tr>
<td>.Vtx</td>
<td>{FAEA5B46-761B-400E-B53E-E805A97A543E}</td>
</tr>
</tbody>
</table>

19. Once all of the updates are made, you will need to restart the Office Search Service. To do this open a command prompt.
20. Run the command net stop osearch.
21. Once the command completes, run the command net start osearch.
22. Now you just need to do a full index update and your new file types should appear.
**Chapter 4: Configuring and Administering Search**

**Maximum Crawl Size**

By default, SharePoint will only index the first 16 MB of a file. After that point, you will get a warning in your crawl log stating “The file reached the maximum download limit. Check that the full text of the document can be meaningfully crawled.” To increase this limit, you need to make a registry change. Be prepared though, you will need to reboot the server to get the changes to take effect.

1. Open regedit.
2. Navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office Server\12.0\Search\Global\Gathering Manager`.
3. Create a new `dword` value named `MaxDownloadSize`.
4. Edit the value to be a decimal, and enter the maximum size you would like in megabytes.
5. Now, you will have to reboot the server to get the settings to take effect.
6. Run a full crawl, and your large files should be fully indexed without warning.

**Reset the Search Services**

From the Microsoft Filter Pack instructions, you may have noticed that to reset the Search service you had to open a command prompt and run `net stop osearch` and `net start osearch`. Now you may have been thinking it would have just as easy to go into Central Administration ➔ Operations ➔ Services on Server and click stop and then start. While that may seem easier, it is not a good idea. When you stop the services from Central Administration, you are actually removing the services from the server. So, you will delete your index and lose all of your configuration settings — not exactly ideal. Anytime you need to reset the Office SharePoint Server Search service, run the command `net stop osearch`. If you need to stop the Windows SharePoint Server Search service, run the command `net stop spsearch`. This will save you lots of unnecessary headaches.

**Crawling Case-Sensitive Web Sites**

If, while adding external content sources, you add web sites running on other platforms that use case-sensitive web sites, you may notice some of the content does not index properly. This was addressed in a SP1 HotFix for SharePoint and in the initial release of Search Server 2008, but you will still need to make a registry update to take advantage of the functionality. To do so see the knowledge base article at [http://support.microsoft.com/kb/932619](http://support.microsoft.com/kb/932619).

**Diacritic-Sensitive Search**

By default, SharePoint sets your SSPs to be diacritic-insensitive. This means that search results will be returned for a word whether the diacritic (also referred to as accent) marks match. This setting is per SSP. To set an SSP to be diacritic-sensitive run the command

```
stsadm.exe -o osearchdiacriticsensitive -ssp YourSSPName - setstatus True
```

You can also do the same thing for WSS Search by replacing `osearchdiacriticsensitive` with `spsearchdiacriticsensitive`.

82
Chapter 4: Configuring and Administering Search

Conclusion

If you find yourself wanting to learn more Search and are looking for buttons and knobs to turn with Search, there are a few web sources you should monitor. The Enterprise Search Team at Microsoft maintains a focused blog at http://blogs.msdn.com/enterprisesearch. It always makes for lively reading. Also, if you go to www.codeplex.com/sct, you can find the Search Community Toolkit. There is a collection of community-driven tools that you can download to help you extend Search even further. As always with community tools, I recommend that you load them on a dev or test server before taking them to production. Better to be safe than sorry.

After reading through this chapter, you should be ready to set up, configure, and administer everything Search. The same holds true with learning any of the inner works of SharePoint; don’t try to learn everything at once. You have this nice big chapter in this book to refer to, as often as you need to, so why memorize it? I promise there will be no tests. Instead take the knowledge and apply it as situations arise.
With previous versions of SharePoint, connecting to, utilizing, and searching systems for which SharePoint did not ship protocol handlers was very hard. You either had to write custom web parts, write custom protocol handlers, build a web application front end, and crawl that or do some combination of all three. Many times the systems you wanted to crawl, but had to write custom components for, were structured systems, such as databases, or line-of-business systems, such as SAP or Siebel. The goal was to bring together your unstructured data and your structured data to make it easier for your end users to be able to search for their information regardless of where it resides.

With the introduction of SharePoint 2007, the process of connecting to non-SharePoint systems has been made much easier with the new Business Data Catalog (BDC). The BDC is a metadata mapping and system registry system that allows you to describe how to connect to your systems, the security involved, and the correct ways to get data. Once the systems are registered, SharePoint can either pull data directly from the system into SharePoint lists, the BDC web parts, or user profiles, or it can index and search the data. This chapter will step you through the architecture of the BDC, how to connect it to your systems, how to use the systems in SharePoint, and how to write programs that use the BDC object model.

**BDC Architecture and Benefits**

The BDC architecture is composed of the BDC metadata model and database, the BDC runtime, and the BDC web parts. The BDC metadata model and database describe the way to connect to the source systems including how to query the system, custom actions, and generic methods. The BDC runtime is implemented using the Shared Service Provider model and provides the runtime capabilities. The BDC web parts talk to the runtime and allow you to display the data from the BDC systems. Building on the web part framework, the BDC web parts can be connected to other web parts to allow you to personalize the data being returned or to send the data from the BDC to other web parts.
Chapter 5: Searching LOB Systems with the BDC

When it comes to Search, the SharePoint search engine has a built-in protocol handler to crawl BDC data and place it in its index. The BDC data then can be searched and have its relevancy ranked just like any other crawled data in the index. Rather than attempting to open up a document when you click on a BDC result, SharePoint will open a BDC profile page, which you will see later, that you can define and that will display the BDC data as well as other data you desire.

The BDC architecture is not just built for searching. Other SharePoint components use the BDC to enhance their functionality. For example, the SharePoint User Profile supports pulling from the BDC. This allows you to use external data in your user profiles. For example, you may want to display the cost center from SAP or sales account from Dynamics or Siebel.

The presentation of your BDC results can expand beyond just search results. You can display business data in lists. SharePoint allows you to define a column using the BDC that can display the data and its associated data from the BDC. Please note that the data from the BDC will be read-only, so you will not be able to modify the data.

Finally, we will explain terminology, since you will see a lot of different terms thrown around in this chapter. The terminology describes the elements of the architecture and programming the BDC, so it is important to understand. The BDC is made up of entities. The easiest way to think about entities is to envision them as abstract objects, like customers, orders, or sales. The entities have properties, relationships, and actions; for example, customers are related to orders, orders, have line items, and line items have prices. As for customers, you can create them, delete them, change them, or place orders for them as actions on them. Some of the other terms, besides entities, that you will see are LOB systems, which are line-of-business systems; and identifiers, which are unique identifiers.

Figure 5-1 shows the BDC architecture.
The Application Definition File

The most important part of the BDC is the application definition file. This is where you define your metadata for the BDC. The BDC supports two types of application definition files. One is a model file, and the other is a resource file. The model file contains the base model for your system. The resource file contains your localized information, including names, properties, and permissions. We will look at the model application definition file, since that is what you will primarily spend your time creating.

XSD Schema File

Since the BDC application definition file is just an XML file, there is an associated Extensible Schema Definition (XSD) schema file for it to make it easier for you to create these files. You can find the BdcMetadata.XSD in your Bin directory for SharePoint under %ProgramFiles%\Microsoft Office Server\12.0\Bin. If you use Visual Studio to create your XML, you can copy the XSD to the working folder for your project and set the SchemaLocation attribute to point to the location to which you copied the file. Visual Studio can then provide IntelliSense for creating your XML. You can also have Visual Studio refer to the file so that you do not have to copy it to your project folder (if you do not want multiple copies of the XSD file).

BDC Definition Editor Tool

Rather than writing BDC definitions by hand, the SharePoint SDK includes a BDC Definition Editor tool. The tool provides a GUI interface to creating BDC files. With the BDC editor, you can connect to databases or web services, create entities and associations, test your code, and import and export your application definition files. I highly recommend that you learn how to use the editor, even if you just use it as a starting point for creating your definition files and then edit them by hand. Figure 5-2 shows the BDC Definition Editor tool.

There is also a third-party tool called the BDC MetaMan. This tool is not free like the SDK tool, so you will have to evaluate whether you want to buy the tool and whether the SDK tool is good enough for the work that you need to do. If the SDK tool does not meet your needs, there is another tool available that may be able to help you to be more productive with the BDC.
Chapter 5: Searching LOB Systems with the BDC

**BDC Metadata Model Overview**

In the following section, we will go through the application definition file in detail so that you understand the different XML elements. Figure 5-3 shows the metadata model for the BDC. We'll step through each of these elements.

![BDC Definition Editor tool](image-url)

*Figure 5-2: BDC Definition Editor tool*
Chapter 5: Searching LOB Systems with the BDC

MetadataObject Base Class

The abstract base class for the BDC objects in the metadata model is the MetadataObject base class. This class provides some default properties that all objects that inherit from it also contain. These properties include the Name, DefaultDisplayName, IsCached, LocalizedDisplayNames, and Properties property. The names are self-explanatory. There is a hierarchy for the displayed names of the objects so that the BDC will first attempt to pull the objects by using the localized display names. If there are no localized display names, then the BDC will pull from the default display name. If there is not one of those set, then the BDC will pull from the Name property. The Name property is your programmatic name and should be immutable, since you want the identifier to be constant even if the display name changes over time.

LobSystem

The LobSystem is the top-level container for your BDC metadata. Everything that you define will begin with the LobSystem. Your LobSystem must have a unique name and can only be defined as a database or web service. Your LobSystem also supports versioning by allowing you to put in a version number. It’s a best practice to update the version when you update your definitions. The versioning supports both minor and major versions, and you should keep backup copies of the older versions of your definitions. One thing you may want to do is use SharePoint to store your definitions, since it supports document versioning. The example below shows a definition for a LobSystem element. The details about the different attributes are below in the tables as well.

```
<LobSystem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://schemas.microsoft.com/office/2006/03/BusinessDataCatalog
    BDCMetadata.xsd" Type="Database" Version="1.1.0.0" Name="pubs"
    xmlns="http://schemas.microsoft.com/office/2006/03/BusinessDataCatalog">
```

Figure 5-3: The BDC metadata model
Chapter 5: Searching LOB Systems with the BDC

Table 5-1 describes the children of the LobSystem element. Please remember that XML is case sensitive, so you will want to make sure that you get your cases right; otherwise, you will run into problems that may be hard to debug.

**Table 5-1**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associations</td>
<td>Container for the related Association collection, which defines associations between data in the BDC.</td>
</tr>
<tr>
<td>ConnectionManager</td>
<td>Reserved attribute for future use to implement an extensible LobSystem type. Do not use.</td>
</tr>
<tr>
<td>Entities</td>
<td>Container for the related Entity collection, which describes business entities in the LOB.</td>
</tr>
<tr>
<td>EntityInstance</td>
<td>Reserved attribute for future use to implement an extensible LobSystem type. Do not use.</td>
</tr>
<tr>
<td>LobSystemInstances</td>
<td>Container for the related LobSystem collection, which describes the system you want to connect to.</td>
</tr>
<tr>
<td>SystemUtility</td>
<td>Reserved attribute for future use to implement an extensible LobSystem type. Do not use.</td>
</tr>
<tr>
<td>Type</td>
<td>Describes the type of system — Database or WebService.</td>
</tr>
<tr>
<td>Version</td>
<td>Major and minor version of the definition file such as 1.2.</td>
</tr>
</tbody>
</table>

Beyond the children, the LobSystem has properties that you can set. These properties will change based on the requirements of your external systems, since different systems will have different capabilities or syntaxes that you will need to take into consideration. Table 5-2 lists the properties supported by the LobSystem:

**Table 5-2**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WildcardCharacter</td>
<td>The wildcard character to use with the WildcardFilter. By default, this value is *, but you may encounter systems where the wildcard is a different character.</td>
</tr>
<tr>
<td>WildcardCharacterEscapeFormat</td>
<td>This is the escape sequence for the wildcard character. By default, this property is ( 0 ). You will need to change this, depending on how your back-end systems expect wildcards to be escaped.</td>
</tr>
</tbody>
</table>
Beyond the preceding properties, the definition also supports the properties listed in Table 5-3, only for web service connections.

Table 5-3

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WsdlFetchUrl</td>
<td>A required property which is the location of your Web Services Definition Language (WSDL) file.</td>
</tr>
<tr>
<td>WsdlFetchAuthenticationMode</td>
<td>Authentication mode to fetch the WSDL file. By default, this is RevertToSelf, which impersonates the current logged-on user. You could also use Passthrough, which passes the user’s credentials to the back-end system for which you’ll need to have Kerberos delegation enabled. Finally, you can use Credentials or WindowsCredentials, which uses the credentials from the Single Sign On (SSO) service you define using WsdlFetchSsoApplicationID.</td>
</tr>
<tr>
<td>WsdlFetchSsoProviderImplementation</td>
<td>Fully qualified name of the ISsoProvider that will provide the credentials. If you are using SharePoint’s SSO provider, this will be &quot;Microsoft.SharePoint.Portal.SingleSignon.SpsSsoProvider, Microsoft.SharePoint.Portal.SingleSignon, Version=12.0.0.0, Culture=neutral, PublicKeyToken=71e9bce111e9429c&quot;.</td>
</tr>
<tr>
<td>WsdlFetchSsoApplicationId</td>
<td>The application ID of the SSO application that will provide the credentials. Please note that this is the app ID and not the SSO display name.</td>
</tr>
<tr>
<td>WebServiceProxyNamespace</td>
<td>Namespace in which to generate the proxy classes.</td>
</tr>
<tr>
<td>WebServiceProxyProtocol</td>
<td>Protocol used by the web service. By default, this is a value of Soap, which means Soap 1.1. Other values include Soap12, HttpPost, HttpGet, and HttpSoap.</td>
</tr>
<tr>
<td>WebServiceProxyType</td>
<td>Fully qualified name of a type to use to generate the web service proxy. This is useful to use if you have web services that provide their own proxy that you have to leverage. This must be fully typed and placed in the GAC.</td>
</tr>
<tr>
<td>WebProxyServerConfiguration</td>
<td>The proxy server to use to access the WSDL.</td>
</tr>
</tbody>
</table>
Chapter 5: Searching LOB Systems with the BDC

**LobSystemInstances and LobSystemInstance**

The `LobSystemInstances` element is a child element of the `LobSystem` element. Its main purpose is to contain the `LobSystemInstance` elements. The `LobSystemInstance` element allows you to create the connection and authentication to your LOB system. By using properties on the element, you set the different criteria for your connection.

For database systems, the BDC can connect to any datasource that supports certain ADO.NET providers, such as SQL Server, Oracle, OleDB, and ODBC. You will also want to set your authentication. Most times you will use Passthrough or SSO authentication by using the `WindowsCredentials` or `Credentials` settings. The following example code shows connecting to a SQL Server database, and Table 5-4 lists the properties for your database connection.

```xml
<LobSystemInstances>
    <LobSystemInstance Name="pubs_Instance">
        <Properties>
            <Property Name="rdbconnection Data Source" Type="System.String">.
            <Property Name="rdbconnection Initial Catalog" Type="System.String">pubs</Property>
            <Property Name="rdbconnection Integrated Security" Type="System.String">True</Property>
        </Properties>
    </LobSystemInstance>
</LobSystemInstances>
```

**Table 5-4**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticationMode</td>
<td>The authentication mode for the connection. By default, it is RevertToSelf but can also be PassThrough, Credentials, and WindowsCredentials.</td>
</tr>
<tr>
<td>DatabaseAccessProvider</td>
<td>Sets the type of the database access provider. By default, this is SqlServer but can also be Oracle, OleDb, or Odbc. A sample ODBC connection string is Provider=sqloledb;Data Source=servername;Initial Catalog=dbname;Integrated Security=SSPI;*.</td>
</tr>
<tr>
<td>RdbConnection XXX</td>
<td>Anything prefixed with RdbConnection will become a property of the database connection. (If you look at the example above, you will see datasource, initial connection, and other properties.)</td>
</tr>
<tr>
<td>NumberOfConnections</td>
<td>Maximum number of connections to the system. By default this is -1, which means unlimited.</td>
</tr>
</tbody>
</table>

(Continued)
**Table 5-4 (Continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SsoProviderImplementation</td>
<td>Fully qualified name of the SSO provider implementation. If you are using SharePoint’s SSO provider, this will be</td>
</tr>
<tr>
<td>SsoApplicationId</td>
<td>ID of the SSO application that contains the credentials.</td>
</tr>
<tr>
<td>SecondarySsoApplicationId</td>
<td>ID of the SSO application that stores the credentials for the Username and Password filter.</td>
</tr>
</tbody>
</table>

When working with web services, the **LobSystemInstance** element contains a few different properties you can use to define how to connect to your web service. The following example shows you connecting to a web service, and Table 5-5 shows the properties you can define for the web service.

```xml
<LobSystemInstances>
    <LobSystemInstance Name="WebServiceLobSystem_Instance">
        <Properties>
            <Property Name="LobSystemName" Type="System.String">WebServiceLobSystem</Property>
        </Properties>
    </LobSystemInstance>
</LobSystemInstances>
```

**Table 5-5**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebServiceAuthenticationMode</td>
<td>The authentication mode for the connection. By default, it is ReverttoSelf but can also be PassThrough, Credentials, or WindowsCredentials.</td>
</tr>
<tr>
<td>NumberofConnections</td>
<td>Maximum number of connections to the system. By default, this is -1, which means unlimited.</td>
</tr>
<tr>
<td>SsoProviderImplementation</td>
<td>Fully qualified name of the SSO provider implementation. If you are using SharePoint’s SSO provider, this will be</td>
</tr>
<tr>
<td>WebServiceSsoApplicationId</td>
<td>ID of the SSO application that contains the credentials.</td>
</tr>
</tbody>
</table>

(Continued)
Table 5-5 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebProxyServerConfiguration</td>
<td>URL of the proxy server to route requests through.</td>
</tr>
<tr>
<td>SecondarySsoApplicationId</td>
<td>ID of the SSO application that has the credentials for the Username and Password filters, and the HTTP and SOAP headers.</td>
</tr>
<tr>
<td>WebServiceUrlOverride</td>
<td>Overrides the URL in the WSDL file to access the web service.</td>
</tr>
</tbody>
</table>

**Entities and Entity Element**

The Entities element is a container for all your Entity elements. The Entity element is an important element of your definition, since it defines your identifiers, methods, and actions and is the basis for the description for your BDC business objects. It can also define associations between your entities, such as customers related to orders. The following code shows an example of an Entities and Entity element hierarchy. Table 5-6 lays out the children of these elements and the optional properties you can set for the Entity element.

```xml
<Entities>
    <Entity EstimatedInstanceCount="10000" Name="authors">
        <Identifiers>
            <Identifier TypeName="System.String" Name="au_id" />
        </Identifiers>
        <Methods>
            ...
        </Methods>
        <Actions>
            ...
        </Actions>
    </Entity>
</Entities>
```

Table 5-6

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EstimatedInstanceCount</td>
<td>The estimated number of instances of the entity. Clients can use this to change how they display the data. By default, the value is 10000.</td>
</tr>
<tr>
<td>Identifiers</td>
<td>Container for Identifier elements.</td>
</tr>
<tr>
<td>Methods</td>
<td>Container for the Method elements.</td>
</tr>
<tr>
<td>Actions</td>
<td>Container for the Action elements.</td>
</tr>
</tbody>
</table>
Chapter 5: Searching LOB Systems with the BDC

Table 5-6 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Title of the entity. This is used in Search and in the business data web parts.</td>
</tr>
<tr>
<td>__BdcLastModifiedTimestamp</td>
<td>Name of the TypeDescriptor element that in an IDEnumerator returns the last modified date for an entity using a timestamp type. By setting this, you allow incremental crawling of your BDC datasource. You may find that the documentation has a capital S for TimeStamp in the documentation. Do not put a capital S on Timestamp or incremental crawling will not work, and you will always do a full crawl.</td>
</tr>
<tr>
<td>Audit</td>
<td>By default, this property is true, which means that every time the entity’s methods are called, an entry is added to the SSP audit log.</td>
</tr>
<tr>
<td>DefaultAction</td>
<td>This is the Action element that will be used in search results and columns. Usually, this is the View Profile action. This property is autogenerated by the BDC.</td>
</tr>
</tbody>
</table>

**Identifiers and Identifier Element**

The Identifiers element is a container for Identifier elements. The Identifier element provides a unique key for your Entity instances. You can think of these like primary keys in SQL Server. Not every Entity requires an identifier. Only Entities that you want to be able to index, search, and have custom actions on need to have unique identifiers. For example, associated entities that do not need to be searched and only need to surface as related to other entities in web parts do not need an identifier. You can have one or zero identifiers, so do not attempt to add more than one; otherwise, you will get a parsing error for your BDC definition.

The identifier has to be a primitive type, which can be an integer, a decimal, a datetime, or even a Globally Unique Identifier (GUID). Most times, to guarantee uniqueness, you will want to use a GUID. The only thing you have to pass to the Identifier element is the name of the identifier and the type. The following is an example of an Identifiers element:

```xml
<Identifiers>
  <Identifier TypeName="System.String" Name="au_id" />
</Identifiers>
```

In the corresponding TypeDescriptor elements that your Identifier elements name, you should tag those elements with an IdentifierName attribute and put the name that you specified in your Identifier element. If you are creating associations, and the identifier you are retrieving is from a different entity, you will also want to set the IdentifierEntityName attribute on your TypeDescriptor element to the destination entity in order to qualify fully the Identifier. For example, if you are retrieving customers and the products the customers ordered, both customers and
products will be modeled as two different entities in your system, so you will want to qualify fully your identifiers using the method described previously.

**Methods and Method Element**

The Methods element is just a container for the Method element. The Method element defines operations against the back end. One thing to realize with the BDC is that it is primarily a read-only solution. You can use a GenericInvoker element, which we will explain later, to perform operations against a back end, but usually LOB systems have their own methods and business logic to perform these operations and the BDC may bypass these business rules.

The way to work with methods is using the children and properties of the element. The sample below shows a sample Method element. Table 5-7 shows the children of the Method element.

```xml
<Method Name="Find_authors">
  <Properties>
    <Property Name="RdbCommandType" Type="System.Data.CommandType,
    System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
      Text</Property>
    <Property Name="RdbCommandText" Type="System.String">Select "au_id",
    "au_lname","au_fname","phone","address","city","state","zip","contract","UserName"
    from authors where au_id=@au_id</Property>
  </Properties>
  <Parameters>
    <Parameter Direction="In" Name="@au_id"/>
    <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0,
    Culture=neutral, PublicKeyToken=b77a5c561934e089" IdentifierName="au_id" Name="au_id" />
    <Parameter Direction="Return" Name="@authors"/>
    <TypeDescriptor TypeName="System.Data.IDataReader, System.Data,
    Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IsCollection="true" Name="Reader">
      <TypeDescriptors>
        <TypeDescriptor TypeName="System.Data.IDataRecord, System.Data,
        Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="Record">
          <TypeDescriptors>
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IdentifierName="au_id" Name="au_id" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="au_lname" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="au_fname" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="phone" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="address" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="city" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="state" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="zip" />
            <TypeDescriptor TypeName="System.Boolean, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="contract" />
            <TypeDescriptor TypeName="System.String, mscorlib,
            Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="UserName" />
          </TypeDescriptors>
        </TypeDescriptor>
      </TypeDescriptors>
    </TypeDescriptor>
  </Parameters>
</Method>
```
Chapter 5: Searching LOB Systems with the BDC

Table 5-7

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsStatic</td>
<td>The default value of this is true, which means that this method can be executed without an entity instance. If it is false, you have to instantiate an entity instance first and execute off that instance.</td>
</tr>
<tr>
<td>FilterDescriptors</td>
<td>Container for FilterDescriptor elements.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Container for the Parameter elements.</td>
</tr>
<tr>
<td>MethodInstances</td>
<td>Container for the MethodInstance elements.</td>
</tr>
</tbody>
</table>

The properties of the Method element define the dynamic properties for your method invocations. For database systems, there are two properties you want to define: RdbCommandText and RdbCommandType. The text will be your invocation, such as a SQL statement, and the type will either be Text or StoreProcedure. You will combine your SQL statement with the Parameter elements to be able to pass parameters into your SQL text from your BDC definition.

For web services, SP1 of SharePoint 2007 added support for custom HTTP and SOAP headers by using properties on the Method element. Table 5-8 has the properties for the Method element.

Table 5-8

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HttpHeadersUserName</td>
<td>The username to pass in the header for authentication. You will want to use Secure Sockets Layer (SSL) for the connectivity to encrypt the data, and you will need to decode the username, since it will be base 64 encoded.</td>
</tr>
<tr>
<td>HttpHeadersPassword</td>
<td>The password to pass in the header. Use SSL and base 64 to decode the value in your web service.</td>
</tr>
</tbody>
</table>

(Continued)
Parameters and Parameter Element

The Parameters element is a container for the Parameter element. The Parameter element can be both incoming and outgoing and allows you to customize your methods. The official list of supported directions is In, Out, InOut, and Return. You will want to make your Return parameter the last parameter in your list. The following example shows both an In parameter and Return parameter. Notice how you have to define the TypeDescriptor for the different parameters so that SharePoint knows the type of the parameter. The Name attribute should correspond to a value obtained from your back-end system or from whatever datasource the parameter is using. You should also notice that the Return parameter is a data reader, since you will be returning a dataset over which the BDC will iterate.

```xml
<Parameter Direction="In" Name="@au_id">
    <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IdentifierName="au_id" Name="au_id" />
</Parameter>
<Parameter Direction="Return" Name="@authors">
    <TypeDescriptor TypeName="System.Data.IDataReader, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IsCollection="true" Name="Reader">
        ...
    </TypeDescriptor>
</Parameter>
```

Please note that with the database systems, you cannot get return values from SQL stored procedures or SQL parameters, and the BDC only supports returning a single dataset from a database.

FilterDescriptors and FilterDescriptor Element

The FilterDescriptors element is a container for the FilterDescriptor element. The FilterDescriptor element has a number of different types to which you can set your FilterDescriptor. Once you have defined your type of Descriptor, you can set properties to customize your filter. The different types of filters supported by the BDC are Wildcard, Comparison, Limit, User Context, Username, Password, UserProfile, SsoTicket, and LastId. Table 5-9 lists the properties for the FilterDescriptor element:
Table 5-9

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UsedForDisambiguation</td>
<td>By default, this property is false. If it is true, the filter will be used in the business data web parts for matching.</td>
</tr>
<tr>
<td>CaseSensitive</td>
<td>By default, this property is false. If it is true, the business data web parts will tell the user that the filter is case sensitive.</td>
</tr>
<tr>
<td>IsDefault</td>
<td>By default, this property is false. If it is true, this filter is the default filter in the business data web parts.</td>
</tr>
</tbody>
</table>

The following sections describe the different types of filters.

**UserContext Filter**

The UserContext filter appends the current user’s Windows or Forms authentication domain and username to the method call. The following example shows using a UserContext filter to pass along the currently logged-on user’s credentials as a parameter to SQL Server, to limit the records returned to only those to which the current user has access:

```xml
<Properties>
    <Property Name="RdbCommandText" Type="System.String">
        SELECT CAST(count(*) as bigint) as Rights
        FROM authors
        WHERE
        au_id = @au_id and UserName = @currentuser;
    </Property>
    <Property Name="RdbCommandType" Type="System.Data.CommandType, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" />
</Properties>
<FilterDescriptors>
    <FilterDescriptor Type="UserContext" Name="currentuser" />
</FilterDescriptors>
<Parameters>
    <Parameter Direction="In" Name="@au_id">
        <TypeDescriptor TypeName="System.String" IdentifierName="au_id" Name="au_id" />
    </Parameter>
    <Parameter Direction="In" Name="@currentuser">
        <TypeDescriptor TypeName="System.String" AssociatedFilter="currentuser" Name="@currentuser" />
    </Parameter>
</Parameters>
```

**Username, Password, SsoTicket, UserProfile Filter**

These filters all pass security information as part of the method call. The Username filter passes an SSO username as part of the call. The Password filter sends an SSO password. The Username and Password filters work together. The SsoTicket filter sends the SSO ticket. Finally, the UserProfile filter allows you to connect to a user profile property to pass. With the UserProfile filter, you need to set a string property on the filter, which is the name of the profile property to retrieve and pass to the method.
Chapter 5: Searching LOB Systems with the BDC

**Wildcard, Comparison, Limit Filter**

These filters take input from the user and pass it along to the back end. The BDC web parts can display and accept input values for these types of filters. The Wildcard filter does what its name implies: allows you to filter based on a wildcard. Using this filter, you can allow users to filter in different ways, for example, by a letter that is contained in a word or by a word that starts with a certain letter.

The Comparison filter allows you to filter on a condition being met. You can think of this as like a SQL `WHERE` clause. For the Comparison filter, you will need to specify a string child element called `Comparator` to tell the BDC what type of comparison you want. The supported values are `Equals`, `NotEquals`, `GreaterThan`, `LessThan`, `GreaterThanEquals`, and `LessThanEquals`. The code that follows shows a `GreaterThanEquals` example:

```xml
<FilterDescriptors>
  <FilterDescriptor Type="Comparison" Name="FilterDescriptor5">
    <Properties>
      <Property Name="Comparator" Type="System.String">GreaterThan</Property>
    </Properties>
  </FilterDescriptor>
</FilterDescriptors>
```

The Limit filter allows you to limit the number of returned items. This is similar to a SQL Server `SELECT TOP` clause. Once you declare this type of filter, you will need a corresponding `TypeDescriptor` to specify the value for your filter limit. An example follows:

```xml
<FilterDescriptors>
  <FilterDescriptor Type="Limit" Name="MaxRow">
  </FilterDescriptor>
</FilterDescriptors>
...
<TypeDescriptor TypeName="System.String" AssociatedFilter="MaxRow" Name="MaxRows">
  <DefaultValues>
    <DefaultValue Type="System.String" MethodInstanceName="FindProducts">100</DefaultValue>
  </DefaultValues>
</TypeDescriptor>
```

Sometimes you will see documentation that refers to a Range filter. This type of filter was deprecated in the beta builds of SharePoint 2007. You should instead use multiple Comparison filters to achieve the same effect.

**LastIdSeen Filter**

The LastIdSeen filter allows you to retrieve chunks of entities from the back end. For database back ends, you probably do not need to do this, since databases support streaming of their records from the back-end system using the datareader. For web services, you may want to use this functionality if you expect that you will get a large number of items back from the web service, which may, in turn, cause timeout or SOAP packet size issues. This filter works with the `IEnumerator`, as you will see later in this chapter.
Chapter 5: Searching LOB Systems with the BDC

Please note that chunking is different from incremental crawling. Both work in the IDEnumerator you create for your entity. LastIdSeen works with chunking; __BdcLastModifiedTimestamp works for incremental crawling. Look at the “IDEnumerator” section for more details.

**Actions, Action, and ActionParameter Elements**

With the BDC, you can allow custom actions on your entities. For example, you may want to allow the user to create a new order for a customer, and to do that you can create a custom action on the customer entity. Other examples include being able to open the entity in its native application. For example, if you are searching Siebel data, you may want to allow the end user to open the returned entity in the Siebel application. To do this, you would provide a custom action.

The Actions element is a container for all your Action elements. The Action element contains the definition for the custom actions you want to perform. The following code example shows the creation of a custom action:

```xml
<Actions>
  <Action Name="Live Search" Position="1" IsOpenedInNewWindow="true"
    Url="http://search.live.com/results.aspx?q={0}" ImageUrl=""/>
  <ActionParameters>
    <ActionParameter Name="Name" Index="0" />
  </ActionParameters>
  <Action>
    <ActionParameters>
      <ActionParameter Name="Name" Index="0" />
    </ActionParameters>
  </Action>
</Actions>
```

As you will notice in the example, you pass in the name of the action that will appear to the user in the user interface, the position of the action if there are multiple actions, whether to place it in the same window or a new one, the URL to call, and any image URLs to display. You will also notice that you can pass in parameters to your Action element using the ActionParameter element. This allows you to customize your actions based on data from the BDC. Here, the name of the entity from the BDC is passed to the action so that Live searches on the name of the entity. You can have multiple parameters that you pass.

Please note that you can only create actions on items with SpecificFinder methods. The reason for this is that with complex types there could be TypeDescriptor collisions. It is not a common case since most times you will give your TypeDescriptors different names, but you should be aware that the BDC will not allow actions on entities without SpecificFinder methods.

**MethodInstance Element**

The MethodInstance element describes how to use your Method objects with the BDC. The easiest way to think about MethodInstance objects and how they relate to Method objects is to assume that one Method may have many MethodInstances. The MethodInstances could have different default values. It’s similar to constructors in code:

The definition of the MethodInstance contains a number of attributes to define the instance. You will find that you need to put in the type of instance and the return values. The following code shows an example of a MethodInstance:

```xml
<MethodInstance Type="SpecificFinder" ReturnParameterName="@authors"
  ReturnTypeName="Reader" ReturnTypeDescriptorLevel="0"
  Name="Find_authors_Instance" />
```
Chapter 5: Searching LOB Systems with the BDC

Table 5-10 lays out the different attributes and a description of each for the MethodInstance element:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Specifies the type of MethodInstance. The possible values are Finder, SpecificFinder, ViewAccessor, GenericInvoker, IdEnumerator, Scalar, and AccessChecker. Please see the information after the table for a description of these different types.</td>
</tr>
<tr>
<td><strong>ReturnParameterName</strong></td>
<td>The name of the return parameter.</td>
</tr>
<tr>
<td><strong>ReturnTypeDescriptorName</strong></td>
<td>The name of the TypeDescriptor object that represents the type that is being returned by this instance.</td>
</tr>
<tr>
<td><strong>ReturnTypeDescriptorLevel</strong></td>
<td>The level of the type descriptor. This is important for complex types where there may be multiple levels in the type descriptor. You should not have to use this if you follow the best practice of not naming your elements the same name.</td>
</tr>
</tbody>
</table>

Since declaring the correct type is important, let’s take a look in a little more detail at the values for the **Type** attribute.

**Finder and SpecificFinder Types**

A Finder returns a set of entities. A SpecificFinder returns a single entity. If you want your BDC data to be used in the BDC List web part, you must implement a Finder method. If you want your data to be searchable and indexed by the SharePoint search engine, you must implement a SpecificFinder method, so that the search engine knows how to crawl and retrieve a single item. If you only want the item to be returned as part of an association with another entity, you do not have to implement any Finder methods.

An example of a Finder method follows. You will notice that the Finder method returns a collection of items from the database.

```xml
<Method Name="Find_Authors_All">
  <Properties>
    <Property Name="RdbCommandText" Type="System.String">SELECT * FROM authors</Property>
    <Property Name="RdbCommandType" Type="System.Data.CommandType, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">Text</Property>
  </Properties>
  <Parameters>
    <Parameter Direction="Return" Name="@Authors">
      <TypeDescriptor TypeName="System.Data.IDataReader, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IsCollection="true" Name="@Authors_RootTypeDescriptor">
        <TypeDescriptors>
          ...
        </TypeDescriptors>
      </TypeDescriptor>
    </Parameter>
  </Parameters>
</Method>
```
Chapter 5: Searching LOB Systems with the BDC

An example of a SpecificFinder follows. As you can see, the SpecificFinder call to the database uses a WHERE clause to return a single record from the database.

<Method Name="Find_authors">
  <Properties>
    <Property Name="RdbCommandType" Type="System.Data.CommandType, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">Text</Property>
    <Property Name="RdbCommandText" Type="System.String">Select 'au_id', 'au_lname','au_fname','phone','address','city','state','zip','contract','UserName' from authors where au_id=@au_id</Property>
  </Properties>
  <Parameters>
    <Parameter Direction="In" Name="@au_id">
      <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IdentifierName="au_id" Name="au_id" />
    </Parameter>
    <Parameter Direction="Return" Name="@authors">
      <TypeDescriptor TypeName="System.Data.IDataReader, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IsCollection="true" Name="Reader">
        <TypeDescriptors>
          <TypeDescriptor TypeName="System.Data.IDataRecord, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="Record">
            <TypeDescriptors>
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" IdentifierName="au_id" Name="au_id" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="au_lname" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="au_fname" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="phone" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="address" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="city" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="state" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="zip" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="contract" />
              <TypeDescriptor TypeName="System.String, mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="UserName" />
            </TypeDescriptors>
          </TypeDescriptor>
        </TypeDescriptors>
      </TypeDescriptor>
    </Parameter>
  </Parameters>
</Method>
Chapter 5: Searching LOB Systems with the BDC

ViewAccessors Element

ViewAccessor allows you to have multiple views of an entity that may have different properties. This is important if you have back-end systems that may return many different views of the same entities. By having a ViewAccessor, you can pull back different fields from your Finder or SpecificFinder methods. You can have none to many ViewAccessors in your definition. Your ViewAccessors will appear in the BDC web parts for users to select from. The easiest way to think about ViewAccessor is to assume that they are a different type of SpecificFinder element. By allowing multiple ViewAccessor elements, you can get around the BDC’s limit of a single SpecificFinder element. To create a ViewAccessor element, you can create a SpecificFinder and change the type to ViewAccessor. In fact, since a ViewAccessor can have a database or web service command in it, you do not even use the same table or method with your ViewAccessor. Usually, you will, but it is not a requirement.

IDEnumerator Element

The IDEnumerator element returns all the IDs from your back-end system, so that the BDC can then use a SpecificFinder to grab each of the corresponding items based on those IDs. Search in SharePoint crawls the BDC first by calling your enumerator and then by calling your SpecificFinder for each item.

If you want to enable incremental crawling of your system, you will want to return a timestamp field in addition to the ID field in your IDEnumerator. You also need to have your back-end system support tracking the modification of its items. For SQL Server, the easiest way to do this is to use a datetime data type and track the modification of your data by updating this column. If you are combining tables in your BDC queries, you will have to calculate in your SQL calls the greater of the two datetime columns and return that one. You will also have to convert your datetime column to the ISO 126 format before returning it to the BDC. An example of the SQL Server CONVERT statement is:

```
SELECT Name, CONVERT(varchar(30), LastModified, 126) FROM Authors
```
For a web service, you will need to make sure that you can handle returning the modified date for items via a method in your web service.

The BDC will take your last modified date and compare its index to its last crawl date and then only crawl the updated information.

The following is an example of using an IDEnumerator with the __BdcLastModifiedTimestamp property. One thing to notice is that you have to declare the __BdcLastModifiedTimestamp property as a string property, but in your TypeDescriptor you must declare the type as System.DateTime.

```xml
<Entity EstimatedInstanceCount="0" Name="entityName">
  <Properties>
    <Property Name="Title" Type="System.String">entityName</Property>
    <Property Name="__BdcLastModifiedTimestamp" Type="System.String">timestamp</Property>
  </Properties>
  ...
  <Parameter Direction="Return" Name="entityReturnParam">
    <TypeDescriptor TypeName="System.Data.IDataReader, System.Data, Version=2.0.3600.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="entityDataReader" IsCollection="true">
      <TypeDescriptors>
        <TypeDescriptor TypeName="System.Data.IDataRecord, System.Data, Version=2.0.3600.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="entityDataRecord">
          <TypeDescriptors>
            <TypeDescriptor TypeName="System.Int32" IdentifierName="ID" Name="ID" />
            <TypeDescriptor TypeName="System.DateTime" Name="timestamp" />
            </TypeDescriptors>
        </TypeDescriptor>
      </TypeDescriptors>
    </TypeDescriptor>
    </Parameter>
  ...
  <MethodInstances>
    <MethodInstance Name="entityIDEnumeratorInstance" Type="IdEnumerator" ReturnParameterName="entityReturnParam" />
  </MethodInstances>
```

For chunking, you will want to have a method in your web service that supports taking an ID and returning results after that ID. From there, you should create a MethodInstance of IDEnumerator. You need to make sure to bind your parameter to your filter. Also, make sure to map your returning parameter to your property, so that the crawler knows which property to call.

**AccessChecker Element**

To allow you to do security trimming at the entity level, the BDC has an AccessChecker MethodInstance type. To learn more about the AccessChecker type, look at the text and sample in Chapter 8.
Chapter 5: Searching LOB Systems with the BDC

GenericInvoker Element

To allow you to call generic methods from your back-end system, the BDC supports the GenericInvoker type of MethodInstance. This type allows you to update data or perform other operations that may not fall into one of the BDC MethodInstance types. Please note that the BDC web parts and lists do not support calling GenericInvoker objects. Instead, you must write to the BDC object model to call these types of methods. The sample below shows a GenericInvoker method:

```xml
<Method Name="AddRecord">
  <Properties>
    <Property Name="RdbCommandText" Type="System.String">
      INSERT INTO MyTable (MyValue) VALUES (@MyValue)
    </Property>
    <Property Name="RdbCommandType" Type="System.Data.CommandType">Text</Property>
  </Properties>
  <Parameters>
    <Parameter Direction="In" Name="@MyValue">
      <TypeDescriptor TypeName="System.Int32" IdentifierName="MyValue" Name="MyValue" />
    </Parameter>
    <Parameter Direction="Return" Name="MyReturnValue">
      <TypeDescriptor TypeName="System.Int32" Name="MyReturnValue" />
    </Parameter>
  </Parameters>
  <MethodInstances>
    <MethodInstance Name="AddRecordMethodInstance" Type="GenericInvoker" ReturnParameterName="MyReturnValue" />
  </MethodInstances>
</Method>
```

Scalar Element

A Scalar returns a single value from the back-end system. For example, you may want to return the total of all customer revenue from your back-end system.

TypeDescriptors, TypeDescriptor, DefaultValue Elements

The TypeDescriptors element is a container for your TypeDescriptor element. A TypeDescriptor element is just what the name implies; it describes the type for your data. TypeDescriptor elements can be complex, in that they can be nested to describe very complex abstract data structures for your BDC datamodel. Plus, you can create TypeDescriptor elements that describe the default values for your parameters inside of the BDC. The easiest way to understand TypeDescriptor is to see an example. The example below shows modeling a database table. Notice that the TypeDescriptor is contained within a Parameter element. You will always set the root TypeDescriptor element’s IsCollection property for a database to true whether you are returning a single record or a stream of records. Table 5-11 explains the schema of the TypeDescriptor element.
Table 5-11

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeName</td>
<td>Specifies the type. This is a .NET type, such as System.String, System.Data.IDataRecord, etc.</td>
</tr>
<tr>
<td>IdentifierName</td>
<td>This is the name of an Identifier that this TypeDescriptor represents and will be either an input or the output, depending on the type of identifier. The BDC will populate the identifier.</td>
</tr>
<tr>
<td>IdentifierEntityName</td>
<td>The name of the entity that contains the identifier this TypeDescriptor describes. This is used in associations and must be set to the destination entity.</td>
</tr>
<tr>
<td>AssociatedFilter</td>
<td>Name of the filter associated with this descriptor.</td>
</tr>
</tbody>
</table>

(Continued)
Table 5-11 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsCollection</td>
<td>If set to true, this descriptor is a collection. You must have one child, TypeDescriptor, that describes an element in the collection. These children can be collections as well.</td>
</tr>
<tr>
<td>DefaultValues</td>
<td>Container for your DefaultValue elements.</td>
</tr>
<tr>
<td>TypeDescriptors</td>
<td>Container for your TypeDescriptor elements.</td>
</tr>
</tbody>
</table>

When working with nullable types, the BDC does support specifying System.Nullable as a type for your TypeDescriptor; however, the BDC web parts do not support this. Our recommendation is that if you return null from your web service, you should model your types as non-nullable types. An example of a nullable type is:

```xml
<Parameter Name="Address" Direction="In">
  <TypeDescriptor Name="AddressType" TypeName="System.Nullable[[CustomerWS.GetAddress, AddressWS]]" AssociatedFilter="AddressFilter">
  </TypeDescriptor>
</Parameter>
```

For the DefaultValues and DefaultValue elements, they describe the default value for your types. This allows you to pass defaults to your different methods. You can have multiple DefaultValue elements that map to multiple MethodInstance objects. You must specify the type of the value, the MethodInstance with which it is associated, and any text that is the value. With date/time values, make sure to pass them in the DateTimeFormatInfo.UniversalSortableDateTimePattern format ("yyyy’-’MM’-’dd HH’:’mm’:’ss’Z’). The example below sets a DefaultValue for a MethodInstance object:

```xml
<DefaultValues>
  <DefaultValue MethodInstanceName="SalesOrderFinderInstance" Type="System.Int32">0</DefaultValue>
</DefaultValues>
```

**Associations and Association Element**

In LOB systems, many times you have relationships among your data. For example, a customer has orders, orders have products, products have producers, and producers have addresses. In order to model these complex relationships between entities, the BDC supports the idea of associations. You can associate a parent and child entity inside of the system and call from the parent or child to get the other, associated, item. One thing to note as you are modeling your entities in the BDC XML is that the XML is parsed from top to bottom. This means that you need to define your entities before you define your associations in the XML. If you don’t do this, you will get an error when parsing your XML.

The Associations element can be added as a child directly below the LobSystem element. It is a container for your Association elements. The example that follows shows an association, and Table 5-12 shows the properties you can set on the Association element.
Chapter 5: Searching LOB Systems with the BDC

<Associations>
  <Association AssociationMethodEntityName="CustomerProducts" AssociationMethodName="GetProducts" AssociationMethodReturnParameterName="Products" Name="CustomerToProduct" IsCached="true">
    <SourceEntity Name="Customer" />
    <DestinationEntity Name="Product" />
  </Association>
</Associations>

Table 5-12

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssociationMethodEntityName</td>
<td>The name of the entity that contains the method that instantiates the association.</td>
</tr>
<tr>
<td>AssociationMethodName</td>
<td>The name of the method that instantiates and traverses the association.</td>
</tr>
<tr>
<td>AssociationMethodReturnParameterName</td>
<td>The name of the return parameter for the destination entity.</td>
</tr>
<tr>
<td>SourceEntity</td>
<td>The name of the source entity in the association. You can have multiple source entities.</td>
</tr>
<tr>
<td>DestinationEntity</td>
<td>The name of the destination entity in the association.</td>
</tr>
</tbody>
</table>

Complete BDC XML Samples

To see some complete BDC XML samples, take a look at the AdventureWorks samples in the SharePoint SDK. The SDK examples show the concepts we talked about throughout the previous section in this chapter, so you can refer to the different BDC concepts defined earlier as you look through the SDK examples.

BDC Web Parts, Lists, and Filters

SharePoint comes with a number of web parts that can work with BDC data. There are five of them that appear in the Web Part Gallery. You will only see these web parts if you have the Enterprise features of SharePoint enabled and activated.

SharePoint also provides the ability to define a column in your list as a business data column. You select which BDC entity to associate the column to, and SharePoint will render the column with all the other columns in your list. You can even retrieve associated data from your BDC entity as part of your BDC column definition. One thing to note is that your BDC column is read-only and has some implications for the document information panel that we will discuss later.

Finally, you can use the BDC as a filter for your SharePoint pages. Just as with other types of filters, you can retrieve BDC data and pass that data to your web parts. From there, you can filter in your web parts the relevant data that the user should see.
Chapter 5: Searching LOB Systems with the BDC

The following section will describe the different BDC web parts, lists, and filters.

**Business Data List Web Part**

The Business Data List web part shows a list of entities from your BDC application. The web part supports sorting, filtering, and custom XSLT to change the rendering for your list. Also, since it is built on the web part framework, you can use web part connections to connect your web parts to other web parts on the page. Finally, you can use custom views to set up default views for your web part for your users. Figure 5-4 shows the business data list web part in SharePoint.

![Figure 5-4: The Business Data List web part](image)

**Business Data Related List Web Part**

The Business Data Related List web part allows you to show the associated entities you have defined in your BDC definitions. For example, if you defined a customer and associated that customer to orders, you would use this web part with the Business Data List web part, so that a user could select a customer and see all the orders for that customer. This web part does not do anything on its own and must be connected to other web parts on the page. Figure 5-5 shows the Business Data Related List web part.
Chapter 5: Searching LOB Systems with the BDC

Business Data Item Web Part

The Business Data Item web part allows you to display a single item from your entity. For example, you could display a single product, employee, or some other data from your back-end system. You can select the view to use for the entity. Furthermore, you could select the item in the tool pane for the web part, but often you will connect this web part via web part connections to the Business Data List web part to display the details for the item currently selected in the list. Figure 5-6 shows the Business Data Item web part in action.
Business Data Actions Web Part

This web part displays all the actions for a particular entity. You normally connect this web part via web part connections to other business data web parts on your page.

Business Data Item Builder Web Part

For every BDC entity, the BDC creates a default profile page that you can customize to display the information from the back-end system and any other data or related information that you may want to display. This web part is only used on the BDC profile pages, since it takes information from the querystring and creates a business data item that it shares with other web parts.
Chapter 5: Searching LOB Systems with the BDC

**BDC in SharePoint Lists**

To support the ability for you to use your business data directly inside of a list, SharePoint has a business data field type. Just as you can create columns of type string or integer, you can create columns of type business data and select the entity from which you want to pull that business data. SharePoint also allows you to pull related properties on the entity into the column. For example, you may want to put an address property from a customer in your list, but if you pull the address, you may also want the city, state, and zip code to appear. In the user interface, you can choose to pull this data into your list as well. Figure 5-7 shows the creation of a business data column in your list.

![Figure 5-7: Creating a business data column](image)

Once you create your column, you can view it in your list. A couple of things to note about business data columns. First, they go directly against your back-end system, so unlike the search engine, which indexes business data and pulls it from the index, the web parts and lists will hit the back-end system. There is some caching of the BDC employees to improve performance, but you should make sure that your back-end system is ready to handle the load if your web pages and lists are viewed on a regular basis. Second, you cannot modify the column value, since it is a read-only value, so if you attempt to view the list in a
Chapter 5: Searching LOB Systems with the BDC

datasheet view, you will find that the cell is read-only. SharePoint includes a Business Data Picker to allow you to pick the entity and the properties on that entity to display. Finally, BDC columns cannot be used in site columns, only in list or library columns. Because site columns are not allowed, you cannot add BDC to site content types. A SharePoint MVP has written a freely available project that adds this capability, which can be found on Codeplex at http://www.codeplex.com/enhancedbdcfield. Figure 5-8 shows business data within a list.

![Figure 5-8: Business data within a list](image)

A quick word about the document information panel and the BDC. If you use the default Document Information Panel (DIP) inside of your office applications, you will find that your BDC data will be rendered in the DIP. You will get a textbox for your Business Data Picker and whatever associated properties you selected will be presented as read-only properties, as shown in Figure 5-9.
One thing to note is that the Business Data Object Picker does not work 100% in the DIP. For example, you can’t search in the Picker, and to resolve a query against the back-end system, you have to enter in the exact key for the value to be found. Many times, you will map the primary key of your database or a unique key in your web service to the BDC. You cannot assume that your users know that the customer ID for the customer they need is 4232123. If the user has already selected the entity using the Object Picker in the browser, then the data and associated values will come down into the DIP. You can then use the metadata from the DIP in your documents. You do not want to try to have your users select other objects in your DIP because of the Picker issues discussed above.

Now, you may be wondering why you just can’t create a custom DIP and provide your own Picker. Well, to create custom DIPs you need to create custom content types, and the business datatype is not available in content types. So, you can’t easily create custom DIPs that use the BDC. You could possibly write to the BDC object model from InfoPath and connect the DIP and the BDC through code, but it is probably not worth the effort.
Modifying Your BDC Profile Page

By default, your BDC entities have a profile page associated with them. This profile page allows users to click the view profile menu item or a BDC search result and see more detail on the BDC entity. The default profile page is just a web part page that contains the Business Data Item Builder web part. This web part takes the identifier from the URL, retrieves the item associated with it, and sends it to connected web parts. The connected web part is the Business Data Item web part. Also, by default, the view profile menu item will appear anywhere BDC appears, such as a list or in the BDC web parts.

Many times, you will want to modify the profile page to customize what is displayed for your business entities. For example, you may want to display related data to the entity, such as sales history, product orders, or other information that extends the default profile. Since the profile page is a web part page, to modify the page is just as easy as modifying any other web part page.

When working with profile pages, you have two choices in the way that you can go about making your modifications. First, the BDC uses a template to create your profile page when you import your application definition file. This template is called _BusinessDataProfile.aspx. When you import your definition, the BDC takes each entity and creates a specific profile page from this template for them. For example, if you had a product and customer entity, the BDC would take the template and create product.aspx and customer.aspx profile page from the template. The profile pages are stored in the content section of your SSP admin URL, such as http://server/ssp/admin/content. If you want to modify all profile pages for all entities, you will want to modify or replace the _BusinessDataProfile.aspx page. When the BDC imports or reimports your application definition, this new profile page will be used for all of your entities. If you are replacing the ASPX file with your own file, make sure to name your ASPX the same name as the original. Also, you can use SharePoint Designer to make more extensive customizations than those you can make in the browser when replacing or modifying the default profile page template. To edit the page in the browser, just go to your SSP administration site, under the BDC section, and click the Edit profile page template hyperlink.

Your second option for modifying the profile page is to modify the individual entity pages, such as product.aspx. This will allow you to make changes for a single entity, rather than modifying the profile page for all entities. To edit these pages, you just need to open one of your entities and edit the web part page. You may be wondering what happens when you reimport your application definition, and whether your changes will be overwritten by the BDC. If the BDC sees a file with the existing name, it will increment the filename, so it does not overwrite the existing file. You then will have to modify the new file or copy your old file over the new file for your changes to be recognized.

Figure 5-10 shows a modified BDC profile page.
Chapter 5: Searching LOB Systems with the BDC

Searching the BDC

Once you have your application definitions and profile pages setup, it is time to start making the search engine crawl, index, and search your BDC content. To do this, you will need to perform some steps, such as adding a new content source, mapping your properties, and creating a scope for your BDC data. Let's take a look at the steps involved in getting BDC data added to your SharePoint search infrastructure.

Adding a Content Source for Crawling BDC Data

The first step in searching your BDC data is getting the search engine to crawl and index that data. SharePoint has a built-in protocol handler for the BDC that you can register and configure from the search administrative pages. SharePoint allows you to crawl the entire BDC with a single content source, or you can break out a content source for each application. Which way you choose to do this depends on a number of factors, such as how many BDC applications you have, the size of the back-end data you are crawling, whether you want to administer one or many content sources, or the security you want to apply to the content source crawls. If you have only one or two BDC applications, and they are similar in their size and requirements, you may just want to use a single content source to make your administration easier.
To add a content source, go to the search administration site. From there, click on the content sources’ link and click the Add Content Source link to add a new content source, as shown in Figure 5-11.

![Figure 5-11: Adding a new BDC content source](image)

Type in the name of your content source such as Customers or Products. Select the Business Data radio button, and under the Applications section, you can either choose to crawl all applications or select individual BDC applications to crawl. Make sure to select Start full crawl on this content source if you want the crawl to begin immediately, or you can trigger the full crawl at some later point. Before you can completely finish your BDC search configuration, you will have to perform a full crawl on your BDC datasource. Make sure, by looking at the search logs, that the full crawl has been completely successful. The first couple of times you do this, you may find security, connectivity, or other types of issues that you will need to rectify before the search engine can crawl your BDC content.
Mapping Crawled Properties

When working with datasources, the SharePoint search engine can uncover new properties in those datasources. In order for those new properties to be used in the search user experience, you have to map those properties to managed properties in SharePoint. You can either map these newly discovered properties to existing managed properties, or you can create new managed properties for your crawled properties. To create a new managed property, you should click on the New Managed Property hyperlink under the Managed Properties section in your search settings. From there, you can map a BDC crawled property to your managed properties. You will normally find that your crawled properties follow the naming pattern of the BDC appname.entity.property, such as pubs.authors.au_fname. Once you have your mapping done, you must perform a full crawl of your datasource in order for SharePoint to complete the operation. Figure 5-12 shows mapping a BDC property to a managed property.

Create a Search Scope

As part of your use of the BDC, you will want to create a search scope for your BDC data, since SharePoint excludes BDC data from the All Content search scope. You can create a search scope that targets a particular content source, which, in this case, would be your BDC content source. To create a new search scope, under the Scopes section of your search settings, you add a new scope. You can use
Chapter 5: Searching LOB Systems with the BDC

the default search results page as part of your search results, or you can specify a new page if you want to customize how the search results look from the BDC. Once you have created your scope, you will want to add a new rule that includes your BDC content source. Figure 5-13 shows adding this type of rule to SharePoint.

![Figure 5-13: Adding a rule for BDC content](image)

SharePoint Designer and the BDC

Before we take a look at the BDC API, we should look at how you can leverage the BDC in your SharePoint Designer (SPD) applications, since writing SPD applications is faster than writing code to work against the BDC. SPD has full support for the BDC as part of its Data View web part, also referred to as the DataView control. Adding a connection to the BDC in SPD is as easy as inserting a new DataView control on your web page. Then, you connect your DataView to your BDC data by adding a new view in your datasource library under the BDC section, as shown in Figure 5-14.
Once you add the BDC datasource, you can insert the BDC data into the application you are building in the SPD. With the DataView control and BDC data, you can do conditional formatting, filtering, sorting, grouping, and even web part connections. One thing that you cannot do is insert, delete, or modify the BDC data, since the back-end BDC system does not support this. Figure 5-15 shows BDC data in a DataView control in SPD. Once you are done configuring your BDC connectivity, you can preview and deploy your new application right from the SPD.
Chapter 5: Searching LOB Systems with the BDC

The BDC API

The BDC includes a rich API for you to develop your metadata models, as well as work with the BDC runtime in your applications. In the interest of space and focus, we will not spend a lot of time looking at the object model for administering the BDC, since you have tools and existing applications that can do many of these functions. Instead, we will spend our time looking at writing to the runtime that you would use if you were building custom applications that connect to the BDC.

The BDC Assemblies

There are two main .NET assemblies that contain all the BDC-related classes. The first is Microsoft.Office.Server.dll and the second is Microsoft.SharePoint.Portal.dll. You will find, when you are working with the BDC assemblies and classes, terms such as Office Server and Application Registry. The codename for SharePoint was Office Server and the codename for the BDC was Application Registry. So, these names are holdover names from the codenames for the product and feature. Wherever you see the codenames, just think of replacing them with the released name in your mind. You will find the assemblies that you need to add to your projects under the ISAPI section of the 12 hive – Program Files\Common Files\Microsoft Shared\web server extensions\12\ISAPI. You will find that you will also use the Microsoft.SharePoint dll in your BDC applications. This is the
Chapter 5: Searching LOB Systems with the BDC

main SharePoint dll for things like connecting to your sites, lists, libraries, and other primitives in SharePoint.

The Microsoft.Office.Server object model is the main object model you will use to write code to interact with the BDC. It contains a number of namespaces that allow you to create application definitions, work with the BDC runtime, connect your back-end systems, and retrieve data, and perform other common operations. The Microsoft.SharePoint.Portal object model will provide some exceptions and connectivity that you will use. The following section describes the Microsoft.Office.Server namespaces that you will generally use for your applications.

**The Microsoft.Office.Server Namespaces**

You will find the BDC specific namespaces under the ApplicationRegistry namespace in the Microsoft.Office.Server.dll. The Microsoft.Office.Server.ApplicationRegistry/Administration namespace is the namespace you will use to write administration type applications with the BDC. With the classes in this namespace, you can create, read, update, or delete the application definition metadata, configuration, or management information for your BDC applications.

The Microsoft.Office.Server.ApplicationRegistry.Infrastructure provides the connection management and security services for the BDC. The MetadataModel namespace, as the name implies, allows you to work with the metadata model. The classes in this namespace are targeted at client applications that talk to the BDC, such as custom web parts, rather than administration-style applications that manipulate the metadata model to make changes to the model. The Runtime namespace provides the classes that allow you to interact with your back-end system, such as getting and setting filters and working with entities. The rest of the namespaces such as the Search, WebService, and others either are for internal use only, or you will find that you do not use them commonly in your code.

**Putting It Together: Building Custom Applications for the BDC**

The quickest way to learn the BDC object model is to write an application that leverages it and then dissect the application. This section will walk you through how to create a web part that connects to the BDC and performs operations on it. We will look at how to get started with the BDC and then drill into the code to understand the components of the BDC object model. Please note that we will not be looking at the object model for metadata management, such as writing BDC application definitions using code. Our recommendation is that you use the BDC tools included in the SDK to create and modify your application definitions. Furthermore, since the BDC definitions are just XML, there are a number of APIs that you can leverage to load, validate, and write out your application definitions, quickly.

The sample application that we will look at will be a simple WinForm application that shows information from the BDC. This application shows you how to connect to the BDC, scan the installed applications, connect to a specific application and its entities, and then work with those entities using the API. Figure 5-16 shows the sample application.
Chapter 5: Searching LOB Systems with the BDC

The first step with any BDC application is to add the right references to your Visual Studio project. You will want to add a reference to Microsoft.Office.Server, Microsoft.SharePoint, and Microsoft.SharePoint.Portal. Then, you will put some namespaces in your code, since you will be using them extensively in your BDC programming. The following code shows these namespaces:

```csharp
using Microsoft.Office.Server.ApplicationRegistry.Infrastructure;
```

The first namespace is for the metadata model for the BDC. This namespace will help us navigate the metadata model and execute methods in the BDC. The second namespace will allow us to connect to and work with the Shared Services Provider database. The third provides us runtime services, such as working with the underlying datasource that the BDC metadata model describes. The final namespace allows us to work with specific back-end systems. In this case, the sample application works with database back ends, but you could also change this to work with web service back ends or just generic back ends.
Chapter 5: Searching LOB Systems with the BDC

Connecting to the Shared Services Database

The first step to working the BDC programmatically is to connect to the Shared Services database from your code. The BDC provides a SqlSessionProvider object that allows you easily to connect to the SSP database. Using this object, you call the Instance method, and from the Instance method, you call SetSharedResourceProviderToUse and pass the name of the SSP for which you want to return a provider instance. By calling this code, the context will be established for your BDC objects to be able to work with the SSP you specify. You will see that the method call does not return any objects. The code for this step is:

```csharp
SqlSessionProvider.Instance().SetSharedResourceProviderToUse(textBoxSSPName.Text);
```

Displaying LOBSystemInstances

Once you have connected to the SSP database, you need to enumerate the BDC applications that exist. From there, you can retrieve a specific application instance and then work with that application to perform your business logic. To retrieve the application instances, you need call the GetLobSystemInstances method, which returns a dictionary of BDC application instances. From there, you can enumerate the instances to find the specific instance that you want to work with. For the sample application, we will enumerate the instances and add them to a listbox, so that we can find all the entities in that instance and then all the fields, methods, and method instances on the entities. You will use a foreach loop to enumerate through the BDC applications, using their keys as the enumeration value:

```csharp
NamedLobSystemInstanceDictionary LOBInstances = ApplicationRegistry.GetLobSystemInstances();
foreach (string strName in LOBInstances.Keys)
{
    listBoxLOBInstances.Items.Add(strName);
}
```

Working with Entities

Once you have your BDC applications, you can retrieve the entities in those applications. The BDC provides a method that returns a dictionary object of entities in your LOBSystemInstance. You will see this is a common pattern with the BDC programming, where you have a method that returns a dictionary of objects that you can enumerate using their keys. To make the programming easier, you will see that the following code loads the LOBSystemInstance object into a private variable that will be used throughout the application’s method calls. The GetEntities method returns a NamedEntityDictionary object that you can enumerate using the Values property on the entities contained in the dictionary. The LOBSystemInstance also has other methods to get properties, localized names, and data classes associated with the LOBSystemInstance you specify.

```csharp
//Load the LOBInstance variable
LOBInstance = ApplicationRegistry.GetLobSystemInstanceByName(_strName);
NamedEntityDictionary LOBEntities = LOBInstance.GetEntities();
foreach (Entity LOBEntity in LOBEntities.Values)
{
    listBoxLOBEntities.Items.Add(LOBEntity.Name);
}
```
Working with an Entity – Finders, Fields, and Methods

Once we have the entities in our application, we can work with the finders, fields, and methods of that entity. For example, you may want to call the SpecificFinder method on an entity or just a standard Finder method to return all the products, orders, or customer names from your back-end system. The code that follows shows how to work with these methods in the BDC. The first thing you will notice is that we get the specific entity by using its name. Then, we get the FinderView on the entity. This method returns to us the view of the entity when the Finder method is executed. From there, we can retrieve the fields in the view and display those. The code then goes through and retrieves the methods and method instances for the entity. You may be wondering about the difference between a method and a method instance. Well, remember that the BDC is an abstraction from your back-end system. While the method may define the action you want to perform, such as getcustomers or getproducts, you need to run a method on the instance of the entity, which is what the method instance represents. You can have one method with many method instances. You can get method instances either from the methods themselves or from the LOBSystemInstance.

```csharp
LOBEntity = LOBInstance.GetEntities()[_strName];
EntityView = LOBEntity.GetFinderView();
foreach (Field EntityField in EntityView.Fields)
{
    listBoxLOBEntityFields.Items.Add(EntityField.Name + " Type: " +
    EntityField.TypeDescriptor.TypeName); 
}

//Load up the Methods
NamedMethodDictionary EntityMethods = LOBEntity.GetMethods();

foreach (Method EntityMethod in EntityMethods.Values)
{
    listBoxEntityMethods.Items.Add(EntityMethod.Name);
}

NamedMethodInstanceIdDictionary EntityMethodInstances =
LOBEntity.GetMethodInstances();
foreach (MethodInstance EntityMethodInstance in
EntityMethodInstances.Values)
{
    listBoxMethodInstances.Items.Add(EntityMethodInstance.Name);
}
```

Executing a Method and Displaying the Results

Once you have your method instances from your application, you can execute those method instances. To execute the method instance, you call the Execute method on the entity. There are a couple of parameters that you need to pass to this method. First, you need to pass a reference to the
MethodInstance entityMethodInstance = LOBEntity.GetMethodInstances() [listBoxMethodInstances.SelectedItem.ToString()];

Object[] parameters = entityMethodInstance.GetMethod().CreateDefaultParameterInstances(entityMethodInstance);

IEntityInstanceEnumerator entityInstanceEnumerator = (IEntityInstanceEnumerator)LOBEntity.Execute(entityMethodInstance, LOBInstance, ref parameters);

Once we have called our method and retrieved our enumerator, there are a number of different ways we can parse through the results. The first way is to grab our entity instances and have the BDC return all the entities as a ADO.NET datatable object. From there, we can use standard ADO.NET programming to create a datarearer and scroll through our records. The following code grabs the entities, parses them with an ADO.NET datarearer, and then creates a string for outputting the entities:

string outputString = "";
System.Data.DataTable dtForGrid = new DataTable();

while (entityInstanceEnumerator.MoveNext())
{
    DbEntityInstance entityInstance = (DbEntityInstance)entityInstanceEnumerator.Current;
    System.Data.DataTable dt = entityInstance.EntityAsDataTable;
    dtForGrid = entityInstance.EntityAsDataTable;
    using (System.Data.DataTableReader dtReader = new System.Data.DataTableReader(dt))
    {
        while (dtReader.Read())
        {
            for (int i = 0; i < dtReader.FieldCount; i++)
Chapter 5: Searching LOB Systems with the BDC

Our next option is to take the data and display it in a **DataGridView** control, rather than in a simple listbox. You will notice in the code that I need to rehydrate the enumerator. It will not be populated, since we have already parsed through it with our previous code. You will also notice that we turn on the **AutoGenerateColumns** properties on the **DataGridView** control, so that the columns have the correct headings for our data. Using the **DataGridView** control is very powerful in your custom applications. You could also connect events from the control to calling methods in your back-end system. For example, if you wanted to write a **GenericInvoker** that modified the back-end data when someone edits the data in your grid, you could do that with your code.

```csharp
Parameters = EntityMethodInstance.GetMethod().CreateDefaultParameterInstances(EntityMethodInstance);

EntityInstanceEnumerator = (IEntityInstanceEnumerator)LOBEntity.Execute(EntityMethodInstance, LOBInstance, ref Parameters);

while (EntityInstanceEnumerator.MoveNext())
{
    if (dtForGrid != null)
        EntityInstanceEnumerator.Current.EntityAsDataRow(dtForGrid);
    else
        dtForGrid = EntityInstanceEnumerator.Current.EntityAsDataTable;
}

dataGridViewReturnedResults.AutoGenerateColumns = true;
dataGridViewReturnedResults.DataSource = dtForGrid;
```

**Working with Associations and Actions**

Even though the sample doesn’t show this, you will want to know how to work with associations and actions in your code. Associations are pretty straightforward, since working with them is similar to the way we worked with other methods that return dictionary objects. The way to retrieve source and destination associations is to call the **GetSourceAssociations** and the **GetDestinationAssociations** methods on your entity objects. Both will return a **NamedAssociationDictionary** that you can parse using the techniques we described earlier.

You can retrieve the actions for an entity by calling the **GetActions** method, which returns a **NamedActionsDictionary**. From there, you can get your actions. The properties you will want to look at are the **GetActionParameters**, which will return the parameters for the action, and the **Url** property,
which gives you the URL to call for the action. If you have an entity instance, you can call the `GetActionUrl` method, which will return the parameterized action URL with the properties from the entity filled in, in the URL.

**TroubleShooting the BDC**

Now that we have seen the architecture, metadata model, and API for the BDC, we can look at troubleshooting the most common issues that people run into when working with this technology.

The first gotcha is that you need to make sure that your entity identifiers are unique. To guarantee this, leverage the unique identifiers from your database or web service systems. The BDC will not work if you do not have unique identifiers.

Second, the metadata is case sensitive. One little capital here or lowercase there can be very hard to track down and debug, so make sure that you are using a consistent model for your capitalization in your metadata.

Third, always test your queries or method calls outside the BDC and make sure that they work. If you are working with a database, try the query in the database tool such as SQL Server Management Studio. If you are working with a web service, try to call the web service from a test application. If the query or method works outside the BDC, it should work in the BDC.

Fourth, use more entities, not less. You may want to try to create complex entities by performing crazy SQL joins or even distributed queries in your SQL statements. While the BDC will allow you to do these sorts of things, you are better off creating primitives using BDC and then using associations to connect your entities. The BDC web parts will work better this way than if you try to parse complex entities.

With associations, you need to make sure that the association is on both entities and that you do not have any filters on your method, since there are differences between entity methods and association methods. Make sure that for your association you are adding the attribute `IdentifierEntityName` in addition to the `IdentifierName`, so that the BDC can fully qualify your metadata definitions. Also, association methods need to be on the source entity, not the destination.

When working with the API, you may find that your connections stay open even after your code completes. You will want to make sure that you close all connections to your back-end systems in the API. You will want to call the `Close` method on your objects when you are done working with them.

**Conclusion**

In this chapter, you saw how you can use the BDC to connect to your LOB systems. While Search is a major component of the value of the BDC, the BDC expands beyond Search to allow you to surface your LOB data in web parts or user profiles. With the BDC, you can quickly build composite applications that bring together disparate datasources in your SharePoint applications.
User Profiles and People Search

Microsoft Office SharePoint Server 2007 provides the capability to share information easily about users, their work, and their expertise. Information sharing improves collaboration and helps solve business problems by delivering the right information at the right time. User information is imported into SharePoint from Active Directory, an LDAP directory service, or any line-of-business (LOB) application, through the use of the Business Data Catalog (BDC) feature in SharePoint Server. This content is stored inside a SQL Server database and is accessible through the user profile. The user profile is governed by security policies which define who can see what information. SharePoint Server search utilizes the user profile information to supplement the search experience to provide people search capability. We will begin the chapter with a discussion of the details and features of user profiles, and then progress to using the profile information from inside of People Search. We will also show how People Search can be extended by adding custom profile properties to the query and results portions of the search process.

User profiles and people search are not available in Microsoft Search Server 2008, and the Business Data Catalog feature of SharePoint Server is only available with the Enterprise Client Access License.

User Profiles

Most organizations store people information (e.g., Department, Phone, Email) in their Active Directory database; another directory services database such as a Lightweight Directory Access Protocol (LDAP) database; or a database such as SQL Server. Additionally, other pertinent information (such as Hire Date, Employee ID) may be stored in line-of-business applications or an HR database. Therefore, a complete and consistent representation of the people in the organization will require data to be aggregated from diverse datasources. Microsoft Office SharePoint Server 2007 is a collaboration platform that allows web sites and content to be personalized with people information, so a rich user profile can help personalize and target content within SharePoint Server, as well as help your users work more effectively together. SharePoint Server can also utilize
people information to provide people-relevant results while users implement search queries. The aggregated user information represents a profile of the user.

People information is made available in SharePoint Server by importing the data from their respective directories. The imported information is stored in user profiles inside a profile database within SharePoint Server. User profiles can also be customized to include properties not available in directory services. Therefore, user profiles can be used to identify connections and relationships among users, such as people with managers and colleagues in common and web site membership. Relationship information can be used to help facilitate more efficient collaboration with colleagues and across teams. This collaboration includes the ability for users to find each other through user-specific search features. User profile information is much more than just a collection of imported and custom properties. It is the source for properties that can be displayed in the public page of a user’s My Site. A user’s My Site can be configured to display organizational relationship information, their colleagues, and web site and Exchange Server Distribution Group memberships through out-of-the-box web parts. We will discuss the use of My Sites more thoroughly in future sections. Before any of the user information can be utilized by SharePoint Server features, a profile import must be completed.

The import process is configured and implemented through Profile Services, which is the shared service for people and personalization features. Profile Services are managed by a services administrator, which may be the Shared Services Provider (SSP) administrator, or it may be delegated to a different individual, since the Profile Services administrator has additional permissions that are not available to SSP administrators. User information can be imported from different Active Directory forests and from line-of-business applications using the Business Data Catalog. The user profile information and managing Profile Services should be a part of the planning process for deploying and configuring SharePoint Server. Specifically, this includes the topics that are summarized in Table 6-1.

Table 6-1

<table>
<thead>
<tr>
<th>Topic</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Sources</td>
<td>A list of connection sources to be used in the import process, including which accounts to be used by Profile Services.</td>
</tr>
<tr>
<td>User Profile Properties</td>
<td>User profile properties will be based on your existing and any future directory services. Determine which properties are the most relevant across the organization. These will represent your core user profiles. Define which LOB applications contain user-relevant information and the properties that can be mapped to the properties of the directory services. SharePoint Server provides a set of default properties that should be reviewed in order to determine what, if any, additional properties are needed. Property choice lists will also need to be configured. They give Profile Services administrators the ability to suggest acceptable values or limit the values for any property by listing the suggested or approved choices.</td>
</tr>
<tr>
<td>Profile Services Policies</td>
<td>Policies govern the type of information viewable by various types of users in the organization. Specifically, this includes whether the policy is enabled, optional, or required; who can view the information; and whether the user can override the policy. Policies control information for both user profiles and My Sites.</td>
</tr>
</tbody>
</table>
Chapter 6: User Profiles and People Search

Table 6-1 (Continued)

<table>
<thead>
<tr>
<th>Topic</th>
<th>Brief Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finding People</td>
<td>People Search allows users to find other users based on their expertise and roles in the organization. Users can be found based on information contained in the user profile store, or limited to the information stored in the public profiles in the user profile store for the SSP. Users can be found by searching for specific user profile property values. What profile information will be indexed and how it will be made available are topics that need to be defined.</td>
</tr>
</tbody>
</table>

We are ready to begin the profile import process. A profile import requires that directory service connections are configured, profile properties are defined and mapped to directory services properties, and profile services polices have been established to govern the visibility of the information. The following section describes the profile import configuration process.

Managing User Profiles

The management of user profiles includes:

1. Configuring the connections that will be used to access data from Active Directory, an LDAP directory, or a LOB application
2. Establishing the properties that collectively will represent the user profile
3. Configuring the mapping between the profile properties and properties native to the directory service or LOB application
4. Performing ongoing indexing of the user profile content

We will begin our discussion with the connections, and how they are configured to achieve a successful property import, and then we will move on to the defining and mapping of the properties.

Profile Services Connections

Profile Services connections are used to define how user-based properties within datasources are imported into user profiles inside of SharePoint Server. Properties are imported into user profiles by connecting to the relevant service or database, and mapping the unique identifier for each property in the shared service to the unique identifier of the corresponding property in the business application.

Imports are performed by the administrator who has access to the SSP administration site and has been assigned the “Manage user profiles” permission.
Chapter 6: User Profiles and People Search

Imported properties can be supplemented from LOB systems using the Business Data Catalog by choosing a relevant entity from the business application, and either mapping that entity to an existing profile property or adding it as a new property. BDC imported properties augment existing profiles; unfortunately, you cannot create or import new user profiles using the BDC. To import from any LOB application, an application definition file (ADF) must be created and imported into SharePoint Server. The process for creating an ADF was provided in Chapter 5, so refer to that discussion if you need to review.

Regularly scheduled imports can be configured to update and maintain the relevancy to all data stores. New connections can be created to support enterprise data requirements that change over time, connections can be removed that are no longer in use, and the properties of existing connections can be modified. After you add or remove connections, or change the settings for an existing connection, you must perform a full import, since user profile information is not updated automatically.

By default, no custom configuration of the import connections is necessary to import from the current domain. To import from an LDAP database, the Business Data Catalog, or resources on multiple servers or across multiple forests, custom connections must be created to each of these sources.

Data cannot be exported using the Profile Services web pages, so the SharePoint Server user profile database cannot overwrite the source databases.

Profile Services connections enable user information to be collected from across directory services and business applications, so that consistent and timely information is always available. Information about users is synchronized across the deployment to all site collections that use the same SSP. Next we will walk you through the process of configuring connections for profile imports.

**Configuring Profile Imports**

The following illustrates the process for configuring import settings. Importing Active Directory data from the current domain does not require the addition of a separate connection or any configuration. A separate connection is required if data is to be imported from an LDAP directory service, or multiple domains or forests, or the Business Data Catalog. Data is retrieved from all connections when an import is launched. Full imports include all data in user profiles, while incremental imports include only the user profiles added since the last import. The process will illustrate importing from Active Directory in the current domain but will refer to resources for an LDAP and BDC connection.

1. From the Central Administration web site, click the link to the Shared Services Administration home page in the Shared Services Administration section of the Quick Launch menu.

2. On the SSP home page, click the User Profiles and Properties link beneath the User Profiles and My Sites section. This reveals the web page shown in Figure 6-1. Review the different options on the User Profiles and Properties page. Notice that there are two sections, Profile and Import Settings, and User Profile Properties. We will focus on the Profile and Import Settings section in the following steps.
Chapter 6: User Profiles and People Search

3. Click the Configure profile import link to reveal the corresponding web page. The datasource for the import and the account used to access the database are configured from this page. A full or incremental import schedule can also be configured.

4. On the Configure Profile Import page, in the Source section, select the source for the import. This is usually the current domain and should be selected by default.

5. In the Default Access Account section, select Specify Account and type a name and password for the access account. It is recommended that you specify an account, rather than relying on the default content access account.
Chapter 6: User Profiles and People Search

6. Scheduling an import is not required but will help ensure that periodic updates are performed, which will help maintain consistency between your data directory and SharePoint Server. Once you decide on the type of import to be scheduled, this import can be configured by using either the Full Import Schedule section or the Incremental Import Schedule section. We will discuss this again later, once we have defined properties and mappings.

7. Click the OK button at the bottom of the page once your configuration is complete.

8. Click View Import Connections from the User Profiles and Properties page. Since we just completed a profile import configuration, an existing connection should exist. Open the existing connection by placing the cursor over the connection name and clicking Edit from the drop-down menu, as shown in Figure 6-2. If an existing connection does not exist, then you will need to create one.

9. Figure 6-3 displays the Edit Connection page. This page allows you to customize the connection to the Active Directory store if necessary. You should familiarize yourself with the options available. You will change the User filter string in the Search Settings section. The existing string, 

\[(objectCategory=Person)(objectClass=User)\]

will import all user accounts, even disabled accounts. If you only want to import user accounts that are active, then the string should be changed to:

\[(objectCategory=person)(objectClass=user)\]

\((!(userAccountControl:1.2.840.113556.1.4.803:=2)))\]. The following knowledge base article discusses making this change: http://support.microsoft.com/default.aspx?scid=kb;en-us;827754.

![Figure 6-2: The View Import Connections web page and the edit option](image-url)
10. Once the filter string has been changed, click the OK button at the bottom of the page. You are now ready to do an import. For a more detailed discussion describing the steps for creating a connection to Active Directory, you can refer to: Add connections to Active Directory, at http://technet.microsoft.com/en-us/library/cc262776(TechNet.10).aspx.

Chapter 6: User Profiles and People Search

11. The user profiles can be viewed by clicking the link titled View user profiles from the User Profiles and Properties page.

12. From the User Profiles and Properties page within the Profile and Import Settings section, click Start full import. You should always ensure that the first import you perform is a full import.

13. From the User Profiles and Properties page, click on the View import log link. The administrator should make a habit of viewing the log after an import has completed. The reader should familiarize themselves with the information presented on the log page.

For creating and managing connections to directory services across Active Directory forests, the reader can refer to “Manage Connections across Domains in Multiple Forests,” at http://technet.microsoft.com/en-us/library/cc262116.aspx.

The purpose of this discussion was to define connections that are used for performing profile imports. At this point, we have not talked about defining the specific properties and the property mappings that are necessary for populating the profile properties with values from the directory store. In practice, we would define properties and their mappings before we complete an import. This discussion provided the reader with the necessary understanding of Profile Services connections, configuring profile import settings, and customizing import connections. The next section discusses profile properties and how to configure mappings and policies.

Profile Properties

The User profiles property data that is imported into SharePoint Server will likely require some definition, mapping, and customization to make the imported data as valuable as possible. User profiles represent two things: a list of users and a set of properties that may or may not contain values for each user. The list of profile properties will need to be configured and managed on an ongoing basis. All user profiles contain the same list of properties with different values for each user. Administrators can add or delete properties in this list, and set personalization policies for each property. Users may also edit property values depending on the specific policy setting. Any change to the properties list or an associated policy will be reflected in every user profile. Property values within a user profile can also be edited, but the value is changed for only that individual user profile. During an import, users are added to the profile, and individual properties within each profile are populated with values based on mappings created between the data directory or BDC and the profile property. Keep in mind that during an import, profile property values, mapped to values in the directory service, will be overwritten every time profiles are imported. Additional properties, or custom properties, as they are sometimes called, can be added to the user profile. These custom properties must also be mapped. The list of user profiles and their related administrative actions are all displayed on the View User Profiles page.

User profiles can be added for users not listed in the current connections to directory services. This is accomplished from the Add User Profile page by clicking on the Add user profile link in the Profile and Import Settings section. This information must subsequently be managed manually and is unaffected by user profile imports. All user profiles, whether imported or added manually, can be indexed and made available through searching and through personalization features. User profiles can be deleted, so information about a user cannot be used in personalization features, even though an individual user is a member of a group imported from directory services.
Configuring Profile Properties

The following steps will walk you through configuring profile properties:

1. Open the User Profiles and Properties page, click the View user profiles link, and review the contents of the revealed page.

2. Place the cursor over the first item in the list to reveal the drop-down menu, and review the menu items, as shown in Figure 6-4. You can see that the user’s profile can be edited. Delete the profile and manage the user’s personal site. Keep in mind that, if you use the Edit option, the changes may be overwritten as part of the next import. You can also search for users according to values in specific properties.

3. Click the New Profile button. Review the different properties that are available, and notice the policy that is shown beneath the Show To column. Any profiles created using this approach will not be affected by subsequent imports. You can also create a new profile from the User Profiles and Properties page.

4. Click the Cancel and Go Back buttons.

5. Return to the User Profiles and Properties page.

6. On the User Profiles and Properties page, in the User Profile Properties section, click View profile properties. From this page, administrators can add, edit, organize, delete, or map user profile properties. These are all key functions in managing user profile data. Profile properties can be mapped to Active Directory, LDAP-compliant directory services, and Application Entity Fields exposed by the Business Data Catalog.
Chapter 6: User Profiles and People Search

7. Review the list of properties and the columns of information. Properties are arranged in the order that they appear on the user profile. The Property Name represents the property’s display name, the name that is displayed when the property appears in the public profile and any sites. The actual property name displayed in the user profile itself is the property that is used by web parts, custom applications, and other programming-related elements. This property is often different from the display name. The actual property name is visible when editing an individual property.

8. On the first property in the list, select the Edit item in the drop-down menu to reveal the Edit User Profile Property. There are a lot of options, so review the information as you scroll down the page. Once you have read through all the sections and different options, click the Cancel button at the bottom of the page. Notice that several of the options are grayed out and not available for modifying.

9. From the View Profile Properties page, notice that you can change the order of properties by clicking the arrows in the Change Order column. The Property Type is self-explanatory. The Mapped Attribute is the attribute in a profile import from a directory service to which each profile property is mapped. Properties can contain more than one value as shown by the Multivalue column. The check mark in the Alias column indicates that the property is related to the user’s name and account.

10. Click the New Property button to reveal the Add User Profile Property page. You will notice that this is the same page as shown in step 8, except that you will see that all of the options are configurable, and none is grayed out.

11. On the Add User Profile Property page, type a name for the property in the Name box and the Display Name in the Property Settings section. Remember, the Display name will be displayed to users. Properties also support multiple languages, as you can see by clicking on the Edit Languages button.

Alternative display names can be provided for each language by clicking the Edit Languages button, clicking Add Language, selecting a language from the menu, and then typing the display name in the new language. The name that is displayed depends on the operating system language being used by the user viewing the property.

12. Choose the data type for the property from the Type menu (for example, string, float, or date). In the Length box, type a value for the maximum length of the property (this option will not appear if it is not applicable to the property type). Enable multiple values, if appropriate, by selecting the Allow Multiple Values check box. If you select this option, indicate how values are separated by clicking either Comma: A,B,C . . . or Semicolon: A;B;C; . . . on the Multivalue Separator menu.

You cannot change a property to a multivalued property if it was initially created as a single-valued property.

13. Properties can be assigned a choice list by selecting the Allow Choice List check box. Enabling this option creates a new section called Choice List Settings. If you are not sure if you should provide a choice list, then select the No Defined Choice List, and you will have the option of adding it later. The property will function as a standard property.
A property that was initially created without a choice list cannot be subsequently changed to have a choice list. Therefore, if you think that you may want a choice list in the future, you should select the Allow Choice List check box at creation.

14. Options for the choice list can be defined by selecting Defined Choice List. Users can be enabled to add their own choices by selecting the Allow Users to Add to Choice List check box. A comma delimited list can be imported.

15. The property’s visibility is controlled by the privacy policy. In the drop-down list box inside the Policy Settings section, you will need to choose whether a privacy policy is Required, Optional, or Disabled. The Required setting defines that the property must contain information. With Optional, each user decides whether or not to provide values. Select Disabled to prevent anyone but the SSP administrator from viewing the property or feature.

16. The next step is to determine who can view information about the property. On the Default Privacy Setting menu, click one of the following: Only Me, My Manager, My Workgroup, My Colleagues, or Everyone. My Workgroup limits visibility to the user and all users who report to the same manager. Choose Everyone to share the information with all users who have the Use Personal Features permission.

17. Select the User Can Override check box to allow users to change the default privacy setting. In general, this should not be chosen without careful consideration. Properties can be replicated to every site collection in the SharePoint farm by selecting the Replicable check box. The Replicable check box may be grayed out and not available for selecting. The privacy policy must be set to Everyone, and the User Can Override check box must not be checked to enable the replicate option. Replication occurs during profile imports.

18. Determine if users will be allowed to edit property values by selecting the appropriate check box in the Edit Settings section, as well as where the property will be displayed on the user’s My Site in the Display Settings section. The Show in the Profile Properties Section of the User’s Profile Page option will be grayed out if Only Me is chosen for the privacy.

To display the property in the Profile Properties Section of the User’s Profile page, select Show in the Profile Properties Section of the User’s Profile page. Show on the Edit Details page displays the property on the Edit Details page available from the personal page of My Site. Show Changes in the Colleague Tracker web part displays changes to the property in the Colleagues section of My Site and all other instances of the Colleague Tracker web part.

19. In the Search Settings section, select the Alias check box to treat the property as a username or account name. An alias property will be used when searching for items authored by a user, targeting items to a user, or displaying items in the Documents web part of the user’s personal site. This option will not be available if Only Me is the privacy setting. Select the Indexed check box to ensure that the property is indexed, and it is a part of the People Search scope schema.

20. The Property Import Mapping section is shown in Figure 6-5. The section is used to define the mapping between the profile property and the directory service or data store. The number of choices in the Source Data Connection menu will be determined by the connections you have already created. You can view current connections and create new connections from the View Import Connections page directly off the User Profiles and Properties page. This menu includes the Master Connection for all directory services connections, and each of the connections to entities in the BDC. Click the appropriate data connection for the mapping.
Chapter 6: User Profiles and People Search

21. In the Data source field to map box, select the property in the datasource that is to be mapped to this specific profile property. All properties from SSPs in trusted domains appear in this list. To map to a property in the master connection, select Data source field to map, and then click a property in the menu, or select Enter field to map and type the name of the datasource field. Property values are mapped when the next import occurs.

22. Click the OK button if you have actually created a new property. If not, click the Cancel button when you’re done.

At this point, property definition and mapping have been completed. In the previous section, we discussed configuring connections, so you are ready to do a real profile import. Before you perform the import, it is a good idea to configure an import schedule. Set up an incremental import that will perform a full import the first time it is initiated. A full import can be reserved for “clearing the slate” and starting over. Profile information can also be mapped to BDC entities. Imports from a BDC connection are supplemental, and they populate properties in profiles that already exist. The next section provides an overview of the process for configuring profile properties and connections for a BDC import.

**BDC Supplemental Properties**

SharePoint Server imports data from a master connection by default, as you saw in Figure 6-5, which is typically an Active Directory or LDAP datasource. Once a successful AD or LDAP import has been completed, it is possible to supplement property values by importing data from a BDC datasource. To enable this type of import, a BDC application definition file (ADF) will need to be created and imported into SharePoint Server. An additional import connection will need to be created for the BDC. Additionally, a relationship between the profile properties imported from the master connection and the entity methods in the BDC application definition must be established. This is done with an Identifier field. The master connection populates the identifier field, which is then used by the BDC Finder and
Specific Finder methods. The following discussion reviews the steps necessary for completing a supplemental import of data from a BDC datasource. Please refer to Chapter 5 for the process for creating an ADF, configuring the SharePoint Server environment for BDC access, and uploading to the ADF into the SharePoint Server environment.

**Configuring for BDC Import**

The following steps will configure a BDC Import:

1. Create an ADF and import the file into SharePoint Server. Test the successful creation and import using the out-of-the-box BDC web parts. This will verify the correctness of the Finder and Specific Finder entity methods.

2. Create a new profile import connection. Select View Import Connections from the User Profiles and Properties web page. Click the Create New Connection button.

3. Choose the Business Data Catalog option from the Type drop-down menu in the Connection Settings section.

4. Define a name for the connection and enter it in the Connection Name field. This name will be displayed in the Property Import Mapping section as one of the options for the connection source. We will call this Business Data Catalog (BDC).

5. Choose an entity using the Business Data Catalog entity picker by clicking on the open book icon located to the right of the Business Data Catalog Entity field. The entity picker should be populated with those entities defined in your application definition files.

6. Keep the default condition Connect User Profile Store to Business Data Catalog Entity as a 1:1 Mapping enabled, since you will be using a single-valued property. If you were importing data into a property that can have multiple values from an entity that does not have associations, then you would select the 1:many mapping option.

7. Choose the appropriate identifier from the drop-down menu to populate the Return Items Identified by this Profile Property option. This identifier is typically a numeric value, but any data type will work. This completes the connection configuration, and an example is shown in Figure 6-6. From the figure, you’ll see that the property EmployeeID was chosen. Each user profile receives an EmployeeID property from the master connection. This property value will be used to do a lookup into the BDC data by calling the SpecificFinder on the entity and retrieving BDC-related information. This BDC information will then be mapped to a property within the user property.
8. Create a new profile property called Cost Center, and configure the Property Import Mapping as shown in Figure 6-7. Select the Map to an associated entity field check box, and then click an entity on the Associated Entity menu. On the Data source field to map menu, click the datasource field to map to this property. This configuration populates the Cost Center property from the BDC connection, not the master import connection. The value of Department from the BDC entity will be mapped to Cost Center.

9. Return to the User Profiles and Properties page and complete a full import.
Chapter 6: User Profiles and People Search

This completes the process for supplementing profile property data with information from a BDC datasource. Profile information is available in many ways within SharePoint Server, and one of those specifically is People Search. SharePoint Server doesn’t index Active Directory or LDAP content directly, but this information can be imported into the profile database, and this content can now be indexed. The following sections discuss People Search and its capabilities in more depth.

People Search

Finding the right information is a common business challenge. Many times we need to identify a technical resource or subject matter expert associated with a given technology, rather than technical information about a given subject. SharePoint Server has the ability to search for people with certain knowledge and expertise. People search functionality supplements the capability for finding content, and allows users to find and leverage the knowledge held by other individuals throughout the organization. People search capability is available through The Search Center in SharePoint Server. Windows SharePoint Services and the different versions of Search Server 2008 do not provide the people search capability. We will begin the People Search discussion with the Search Center.

The Search Center

SharePoint Server comes with two different search site templates: Search Center and Search Center with Tabs. The following discussion will focus on the latter. Search Center plays a central role in the user experience with SharePoint Server search by providing a more functionally rich and pleasing interface to the small search box that populates all sites. The Search Center with Tabs (SCT) site can only be included as part of a site collection that has the publishing features activated. An SCT is automatically provisioned as part of the Collaboration Portal site template. Figure 6-8 displays an SCT site that was provisioned from the corresponding site template. We will begin by discussing the People Search web page and the tabbed user interface.

![Figure 6-8: An example of the Search Center with Tabs site](image-url)
Chapter 6: User Profiles and People Search

**People Search Page and Tab**

The SCT site in Figure 6-8 displays two tabs by default: All Sites and People. Each tab includes a query box centered on the page, with an area beneath for displaying search results. The tabbed structure is extensible; existing tabs can be modified, and new tabs can be created by editing the page. An overview of the two different tabs is given in Table 6-2, but we will focus on the People tab.

<table>
<thead>
<tr>
<th>All Sites</th>
<th>This tab is selected by default when a user browses to the SCT site. It is associated with the default.aspx page, which is the default or home page for the SCT site. The search box is configured to work with the All Sites search scope.</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>The people.aspx search page is associated with this tab. The search page contains different web parts and options that are dedicated to people search functionality.</td>
</tr>
</tbody>
</table>

The people.aspx web page contains a Search Box web part that is different from the one on the default.aspx page. Similarly to the search box on the default.aspx page, you can enter and execute keyword queries in this one. However, the People Search box, as the web part is called by default, provides a Search Options pane, when the Search Options link directly to the right of the query text box is clicked on. This reveals the capability shown in Figure 6-9.

The reader will notice that some of headings in the Search Options pane are the names of properties contained inside of the user profile. Therefore, users can be identified according to values contained in the profile. This emphasizes the power of information contained in the user profile, and the flexibility available for populating the profile, as we discussed in the first part of the chapter. The options pane also provides a simpler interface for executing keyword queries. For example, a user could enter Web Designer into the Title text box and execute the query. Upon execution, SharePoint Server would translate the option pane content into the keyword syntax `JobTitle: "Web Designer"`. Some of the properties in the options pane can accept multiple values. For example, notice the lookup button to the right of the Responsibilities and Skills text boxes. The Skills lookup button reveals the dialog box shown in Figure 6-10, displaying all of the possible values that can be entered for the Skills property.
The basis for this is that the Responsibilities and Skills properties have been configured to allow multiple values. This multivalue capability is configured for a property, as we discussed earlier in the chapter. Figure 6-11 shows the Choice List configuration for the Skills property, and you can see that its allowable property values coincide with those shown in the lookup dialog in Figure 6-10.

Results Page

People Search results are displayed on the peopleresults.aspx web page. An example of the results page is shown in Figure 6-12. For a query using the text “Brian”, two results are shown; they are sorted by Social Distance, by default. Social Distance represents a relationship between the person issuing the query and his or her colleagues.
Chapter 6: User Profiles and People Search

Results in Figure 6-12 show that each individual is represented by a picture, their name, and additional information from their user profile. You can find out more information about the individual by clicking the name hyperlink, which will take you to their corresponding personal site. The gray circle to the left of the individual’s name represents the capability to integrate presence information. Presence information is a result of integration of SharePoint Server with Microsoft Communication Server or its equivalent and provides an indication of whether the individual is currently online, and of such capability as instant messaging.

Search results can be further refined by user profile properties such as job title and department. The results can alternatively be viewed based on search term relevance by clicking on the View by Relevance link. Relevance for People Search is based on the profile properties. You can consider a person’s profile as a document about that person with his or her name as the title. The more search query terms that are also present in the person’s profile, the more relevant the result.

You’ll also notice that the results have an Add to My Colleagues link. This is one of the social networking features of SharePoint Server. The profile database maintains a list of your colleagues that is populated during a profile import. Initially, this list is populated with a set of “immediate colleagues” that are computed from your profile properties: manager, peers, and direct reports. You can also add and remove colleagues from your My Site or your My Profile page. By clicking on the Add to My Colleagues link, the Add Colleagues web page is displayed as shown in Figure 6-13.
This page allows you to set the privacy policy for who can view this colleague, and to assign the colleague to an existing or custom group. As a result of following this approach for the individual who was the first result in Figure 6-12, the People Search Results page changes as shown in Figure 6-14.

After adding Brian Burke as a colleague, a search with the text “Brian” will return results showing Brian Burke grouped within My Colleagues and ranked higher than all others (Everyone Else). This is an example of social distance; the grouping of results is directly influenced by the colleagues that I’ve added to my colleagues’ list. Also shown in the search results page are links to Alert Me and RSS. These provide the user with the ability to be notified when changes occur within these search results.
**Customizing People Search**

As mentioned previously, the People Search web part adds the capability of the Search Options pane. This pane enhances the search experience by exposing managed properties that are part of the user profile. In this way, it circumvents the need for the user to know the keyword search syntax and simplifies the ability to do custom searches. The options pane can be hidden by disabling the Show Search Options feature in the Miscellaneous section of the web part’s property pane, as shown in Figure 6-15.

![Figure 6-15: The People Search web part properties pane showing the Miscellaneous section](Image)

Also visible in the Miscellaneous section is the Properties property. By clicking on the button directly to the right of the Properties text box, you will reveal the XML content that governs what properties are displayed in the options pane. This XML is:

```xml
<Properties>
  <Property Name="FirstName" ManagedName="FirstName" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:FirstName"/>
  <Property Name="LastName" ManagedName="LastName" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:LastName"/>
  <Property Name="Department" ManagedName="Department" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:Department"/>
  <Property Name="JobTitle" ManagedName="JobTitle" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:Title"/>
  <Property Name="Responsibility" ManagedName="Responsibility" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:SPS-Responsibility"/>
  <Property Name="Skills" ManagedName="Skills" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:SPS-Skills"/>
  <Property Name="QuickLinks" ManagedName="QuickLinks" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:QuickLinks"/>
</Properties>
```

Notice that each of the Property elements represents one of the properties displayed in the options pane. Therefore, additional managed properties can be added to the options pane to further enhance and simplify the user experience.
Customizing the Advanced Search Options Pane

The properties displayed in the options pane can be modified by editing the XML that defines these elements. Let’s consider that you want to add the Assistant property to the options pane, so that managers can have an easy way to identify who has which assistant.

1. Navigate to the Metadata Property Mappings web page, which is one level below the Search Settings page in Central Administration.
2. Locate the Assistant managed property, and click on the People:Assistant (Text) hyperlink under the Mappings column.
3. From the Edit Crawled Property page, copy the value of the Property Name in the Name and Information section. This value, which is shown here, represents the value for the ProfileURI attribute for the Property element:

   urn:schemas-microsoft-com:sharepoint:portal:profile:Assistant

4. Create a new Property element with the appropriate values for each of the attributes as shown in the following example:

   <Property Name="Assistant ManagedName="Assistant" ProfileURI: urn:schemas-microsoft-com:sharepoint:portal:profile:Assistant" />

5. Insert the new Property element into the XML using the dialog box for the Properties property in the Miscellaneous section in the tool pane.
6. Return to the SCT site, and click on the People tab to make sure that the page is refreshed, and you should see the new property added to the options pane, as shown in Figure 6-16.

   ![Figure 6-16: The new options pane with the Assistant property added](image-url)

The preceding process illustrates how the options pane can be customized to show additional managed properties, so that the search experience can be even more valuable to your end users. We can continue with this same concept and display a profile property, such as Assistant, as part of the results. This is discussed in the following section.
Adding Custom Properties to the People Results Page

1. Open the tool pane for the People Search Core Results web part, and expand the Results Query Options section.

2. Place the cursor inside the Selected Columns property text box to reveal the button directly to the right.

3. Click on the button to reveal the dialog box that contains the XML string that determines the profile property information that is displayed by the People Search Core Results web part. The XML string is shown here with a new Column element added to the bottom of the string. The new Column property called Assistant will be added to the search results.

```xml
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Columns>
        <Column Name="AccountName"/>
        <Column Name="UserProfile_GUID"/>
        <Column Name="PreferredName"/>
        <Column Name="JobTitle"/>
        <Column Name="Department"/>
        <Column Name="WorkPhone"/>
        <Column Name="OfficeNumber"/>
        <Column Name="AboutMe"/>
        <Column Name="PictureURL"/>
        <Column Name="WorkEmail"/>
        <Column Name="WebSite"/>
        <Column Name="Path"/>
        <Column Name="HitHighlightedSummary"/>
        <Column Name="HitHighlightedProperties"/>
        <Column Name="Responsibility"/>
        <Column Name="Skills"/>
        <Column Name="SipAddress"/>
        <Column Name="Assistant"/>
    </Columns>
</root>
```

4. Next you’ll need to edit the XSL so that the Assistant data is displayed as part of the results. The Data View Properties of the People Search Core Results web part will need to be modified. Click on the XSL button to reveal the XSL string.

5. Browse through the XSL and familiarize yourself with the default format. Use Ctrl+A to highlight all of the XSL string, and copy and paste it to a file, so that you can save the original string. Once the original string is saved, scroll to the bottom of the XSL string and locate the following portion. The “office” value has been replaced by the text “assistant” in the lines highlighted in yellow. This will remove the property value for the Office profile property from the display and replace it with the Assistant value.

```xml
<xsl:template name="DisplayOfficeProfile">
    <xsl:param name="title" />
    <xsl:param name="dep" />
    <xsl:param name="phone" />
    <xsl:param name="assistant" />

    <span class="psrch-Metadata">
```

152
Chapter 6: User Profiles and People Search

6. You need to make one additional change to the XSL string. Locate the following portion of the XSL. Replace the "office" text and "officenumber" text with "assistant" as shown in the bold text.

```xml
<div class="psrch-Description">
  <xsl:call-template name="DisplayOfficeProfile">
    <xsl:with-param name="title" select="jobtitle" />
    <xsl:with-param name="dep" select="department" />
    <xsl:with-param name="phone" select="workphone" />
    <xsl:with-param name="assistant" select="assistant" />
  </xsl:call-template>
</div>
```

7. Close the XSL dialog and make sure that you save your changes.

8. Click the OK button to close the web part tool pane.

9. Publish the modified page.

10. Navigate to the Configure Search Settings page and select Reset All Crawled Content. Once the content has been reset, complete a full index of the content. Wait for the index process to be completed before returning to the SCT site.

11. Return to the SCT site and reissue the “Brian” text query from the People Search page. The results are displayed in Figure 6-17. You will notice that the Assistant value for Brian Burke is Adina Hagege, and for Brian Cox it is Megan Sherman. The reader should compare Figure 6-14 and Figure 6-17 to see that the value for the Office property has been replaced by the Assistant value.
Chapter 6: User Profiles and People Search

The last two sections have hopefully illustrated that the search query user interface and the search results display can be easily customized without the requirement of any .NET programming. The source of the simplicity is the use of XML to govern the elements to be displayed.

Summary

Using the people search capabilities of SharePoint Server can very easily unlock a huge amount of tacit knowledge in an organization. It’s often very surprising how this latent information can lead to a more productive and efficient way of working. If you’ve ever thought “I wonder who I could ask about X,” then you need People Search!

This chapter has outlined the steps needed to configure and customize the built-in people search capabilities in SharePoint Server and hopefully opened up a new way for people in your organization to find each other and share knowledge.
Being able to crawl and index content is a core feature of any search engine. SharePoint ships with several indexing connectors out-of-the-box that enable it to connect not only to Microsoft content repositories, such as Microsoft Exchange Public Folders and Windows NTFS file shares, but also to non-Microsoft repositories, such as Lotus Notes, Documentum, FileNet, Web Sites, Structured Data, and others.

Indexing content is a required function for virtually every Enterprise Search implementation; after all, you need to index all of your content in order for your users to find it, right? Well, while that statement is partially true; it’s not the full story.

When SharePoint Server shipped, the search functionality required that you create content sources to crawl and index your data in order for it to be returned in search results. This was the only way to get results in front of users (aside from writing custom web parts or controls that returned results from sources other than the local index), so you were effectively tied to the process of pulling your data across the wire, indexing it, writing it to the index on the index server, and managing security.

This changed for the better when Search Server 2008 was released. Search Server 2008 shipped with a new set of features, focused on the ability to send a user query to multiple index stores (including the SharePoint Server or Search Server local index), and then to federate the results in the Search Center UI. This new feature set opened up a new approach to making results from certain types of repositories available through search in a more efficient and user-friendly way.

In this chapter, we will cover the federation features in detail, including what the federation framework looks like, how to configure and customize a Federated Location, what the new Federation web parts do, and how you can configure them.
Chapter 7: Extending Search with Federation

The federation features found in Search Server are also available in SharePoint Server through the Infrastructure update for Microsoft Office Servers (KB951297), which can be downloaded from downloads.microsoft.com. The federation features will also be included in SharePoint Server Service Pack 2. The federation feature set is identical for Search Server and SharePoint Server after the Infrastructure update has been applied.

The Concept

The first things we cover will help you to understand what federation in SharePoint Server and Search Server is, and, at a high level, how it works, then we can get into the details further on in the chapter.

I will start from the beginning. In the context of SharePoint Server and Search Server, the term “federation” means the ability to send a user query to a remote search engine or line-of-business application and then render the results within the Search Center UI. Now that statement probably raises more questions than it answers, so I’ll start by breaking it down a bit.

All of the federation functionality is centered on a concept called “Federated Locations.” A Federated Location is configured by an administrator through the Search Administration dashboard and contains all of the configuration information needed, both to send a query to the location and render the results. Among other things, it includes a query template URL, which is used to send the query to the location it controls, when a user’s query is sent, based on the query terms matching a configured pattern, the authentication settings for the location, and the XSL required to render the results.

A Federated Location can be configured for any remote source that supports the OpenSearch standard, and that can return results as XML. Your remote source could be another search engine either inside or outside your firewall or an application. For example, search.live.com is OpenSearch compliant and can return results as XML (through its search results RSS feed) and is actually one of the sample Federated Locations shipped out-of-the-box. Another location could just as easily be an application that indexes itself, for example, a CRM application or a database server.

If the location you want to federate to is OpenSearch compliant and can return results as XML (either in the RSS, ATOM, ATOM2, or RDF format), then you can configure everything you need to configure right from within the Administration pages in a handful of clicks and text boxes.

If the location you want to federate to is not OpenSearch compliant or can’t return results as XML (The OpenSearch standard includes returning results in a few formats that the federation framework doesn’t support, such as HTML or XHTML), then you’ll need to write or purchase a Federated Search Connector and configure the Federated Location to point at the connector, rather than at the source.

Once configured, Federated Locations are very easily exported and imported through the UI. When a location is exported, it is saved as XML in a Federated Location Definition (FLD) file and has the .fld file extension. If the remote location doesn’t require a Federated Search Connector to work, such as search.live.com, then importing an exported FLD is all you need to do to create the location. If the location does require a connector, then you’ll need to install, and possibly configure, the connector, and then import or manually configure the Federated Location to point to that connector.
Many Internet search engines and enterprise applications support the OpenSearch standard and are capable of returning XML, making them very easy to connect to SharePoint Server or Search Server. There’s also a growing list of independent software vendors (ISVs) and enterprise application vendors that are supplying Federated Search Connectors for applications that don’t support OpenSearch, enabling Federated Search to connect to a variety of line-of-business applications.

Federated search results are displayed in two new search web parts, a Federated Results web part and a Top Federated Results web part. The Federated Search web part displays a preconfigured number of results from a single connected Federated Location, and the Top Federated Results web part displays a preconfigured set of results (usually just the top one or two results) from a list of up to 10 connected Federated Locations. The Top Federated Results web part will display the results from the first location to return results as it works, in order, down the list of connected Federated Locations with which it has been configured.

Results from a federated location can only be displayed using the Federated Results web parts; it’s not possible to merge local and federated results into a single results set displayed within the Core Results web part. Figure 7-1 shows results from the local index on the left side, and federated results returned from www.live.com along the right side.

Figure 7-1: Federation in action
Chapter 7: Extending Search with Federation

Crawl and Index or Federate?

One point to make very clear before getting too far into this chapter is that federation is not a replacement for crawling and indexing. It’s an alternative approach that has its merits, just as crawling and indexing do. It’s highly likely you will want to employ a combination of both approaches to meet your requirements.

That being said, there are pros and cons for each approach, depending on what your requirements are, and the decision points are generally different for each content source and implementation. There’s no right or wrong answer; which approach is more suitable really does depend on the situation. The rest of this section talks through some of the decision points that you will commonly encounter when trying to decide whether to federate or to crawl and index. These decision points are discussed in order from the most important factors to the least important.

Unified Result Set

When content is crawled and indexed, it ends up in the local index of SharePoint Server or Search Server; this means that the Search service knows about all of the content that it has and can make a sensible decision on how to rank the results of a search query, taking into account the rest of the content in the index. Results from the local index appear in a single relevance ranked result set in the UI; however, this is not true for federated results from other search engines or applications. Federated results are always displayed in separate web parts (one for each location), and there’s no way to combine them into a single result set with results from the local index. For the most part, this works well. For example, if you’re federating to a customer relationship management (CRM) application, it generally makes sense to display “customer” results separately from documents and web pages off your intranet. Sometimes, however, this doesn’t work so well, or at least it doesn’t work the way you might expect. If we take another example, that of federating to another SharePoint Server or Search Server index, results from those sources will also be displayed separately from the core results returned from the local index, which might not be what you think would happen or actually want (although this approach does work well for geo-distributed SharePoint Server deployments where bandwidth and latency issues can hinder crawling and indexing.)

This is the “I want a single unified results set” decision point. Anything you crawl and index, will, by default, be displayed ranked by relevance in a single result set. This isn’t possible with federation; you can’t merge the results from multiple federated locations or the local index into a single result set. Of course, how much of an issue this is for you, depends completely on whether you actually want to merge federated results with federated results or federated results with local index results. For many implementations, keeping results separated makes complete sense, and this is a nonissue. If this is an issue for you for a specific content source, then it’s a deal breaker as far as using federation is concerned, as there’s no way to get around it.

Freshness of Results

Crawling and indexing content provides you with complete control of the freshness of the data in the index. When a new content source is configured, a crawl schedule is typically set. This crawl schedule defines how often incremental or full crawls occur and means that you get to decide how fresh search results are for your users. When you use federation, unless you own the administration of the application or search engine you’re federating to, you’re at the mercy of the administrator that does own it, when it comes to how often the content in the index is refreshed, and what content is indexed. Having
control of when and how content is crawled and indexed does come with a price though; you need to have the right indexing connectors installed and configured for the content source. Possibly, you will need the Business Data Catalog that’s shipped with the Enterprise version of SharePoint Server. There is a high probability that there will be some security work to do to manage the security of the content you index, once it’s in the index, so users don’t get results they shouldn’t have access to.

This is the “I want to control how fresh my results are” decision point. There’s a tradeoff between complete control of how and when a content source is indexed (along with managing that process and the data once it’s in the index), and possibly very little control but much less admin overhead, for using federation.

Avoiding “Double Indexing” Content

Many application servers (such as Database, Email, Archiving and Collaboration tools) have built-in search functionality that’s available through the application’s UI or through APIs. Crawling and indexing these servers generally means that you’ll end up double indexing content if your users are already using the search functionality of the application itself. Furthermore, you’ll need to have the right indexing connectors installed and configured, and possibly have to do some security configuration work to ensure that results are properly security trimmed. This type of application is prime federation fodder. More often than not an application that includes built-in search functionality will understand its data and data structures better than anything else does, and, for this reason, it’s likely to be able to return very relevant results. Using federation in this scenario makes complete sense, provided that you don’t need a unified results set, and that you own the application you’re federating to or are happy with the way it’s configured to index it’s content. This approach also allows you to create a consolidated or aggregated search experience, pulling together several different sources into one place.

This is the “My <insert app name here> already indexes itself, I don’t want to double index” decision point. If the application server supports OpenSearch, or if there’s a Federated Search Connector available for it, then choosing to federate to it, rather than crawl and index it, is an easy decision to make.

Content outside Your Firewall

It’s common for organizations to want to search across data that isn’t necessarily within their own firewall for information, such as competitive intelligence, reference documentation, financial information, and news from Internet based search engines or subscription services. Previously, these sources would need to be crawled and indexed, but if they support OpenSearch, or there’s a Federated Search Connector available for them, it can make much more sense to federate to these external sources, rather than manage the overhead of crawling and indexing them.

This is the “It’s not my data, I don’t really want to crawl and index it if I can help it” decision point, and again, it comes down to how you want to manage the rendering of the data, and whether it makes sense for you to have it in the local index or just show the results from the remote source.

If this section has taught you anything, we hope it’s that both crawling and indexing, and federating have their places and can be used interchangeably depending on your requirements. There are no hard and fast rules about when to use one versus the other.
Chapter 7: Extending Search with Federation

Federated Search Locations

Federated Locations are what drive the federation functionality in SharePoint Server and Search Server. They’re configured through the Administration dashboard through the “Federated Locations” link in the left hand navigation.

SharePoint Server and Search Server ship preconfigured with the three locations described in Table 7-1.

Table 7-1

<table>
<thead>
<tr>
<th>Federated Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Search Results</td>
<td>Returns results from search.live.com</td>
</tr>
<tr>
<td>Internet Search Suggestions</td>
<td>Returns “Related Searches” suggestions from search.live.com</td>
</tr>
<tr>
<td>Local Search Results</td>
<td>Returns results from the local Search Server or SharePoint Server Index</td>
</tr>
</tbody>
</table>

It’s very easy to delete the preconfigured locations if you don’t need or want to use them; just click the name of the location on the drop-down list, and press Delete. If you want to save locations, you can click Export and save the location to an FLD before deleting it, as shown in Figure 7-2.

Figure 7-2: Edit, copy, delete, or export a location
Chapter 7: Extending Search with Federation

Clicking the New Location toolbar button does pretty much exactly what you would expect it to do and displays the Add Federated Location page. Table 7-2 describes the function of each field on the page, and, if you follow along, entering the values in the Example column, you’ll end up with a fully functional OpenSearch 1.0/1.1 Federated Location configured to return results from TechNet search on http://search.technet.microsoft.com/search. The sections after the table explain several of the fields in more detail.

Table 7-2

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location Name</td>
<td>The location name is used internally and must be unique. It cannot contain any spaces or extended characters, and it can’t be changed through the UI after it saved. It is a required field.</td>
<td>“TechNet”</td>
</tr>
<tr>
<td>Display Name</td>
<td>The display name is the name that’s displayed in the web part tool pane when a Search Center administrator configures a Federated Search Results web part. It can contain extended characters and spaces, and it’s a required field.</td>
<td>“TechNet Search”</td>
</tr>
<tr>
<td>Description</td>
<td>The description is also displayed in the web part tool pane when a Search Center administrator configures a Federated Results web part. It can contain extended characters and spaces, and it’s a required field.</td>
<td>“Returns results from Microsoft TechNet Search”</td>
</tr>
<tr>
<td>Author</td>
<td>The author field allows you to tag the name of the person creating the location into the location. It’s for administration only and isn’t functionally used by Search service. It’s an optional field.</td>
<td>“Your name here”</td>
</tr>
<tr>
<td>Version</td>
<td>The version number allows you to keep track of changes to the location. It’s for administration only and isn’t functionally used by the Search service. It isn’t automatically incremented, so if you want to use the field, you’ll need to increment the number manually every time you make a change to the location. It’s an optional field and requires numbers and at least one period.</td>
<td>“1.0”</td>
</tr>
<tr>
<td>Trigger</td>
<td>Triggers determine when the location is sent a user’s query terms. They’re an important concept to understand, and there’s a section on them later in this chapter with much more detail.</td>
<td>Click the “Always: Query should always match” radio button.</td>
</tr>
</tbody>
</table>

(Continued)
### Table 7-2 (Continued)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location Type</td>
<td>The Location Type in the Location Information group allows you to configure the location as an “OpenSearch 1.0/1.1” location or a “Search Index on this Server” location. This setting controls how the query is sent to the location and is covered in more detail later in this chapter.</td>
<td>Click the “OpenSearch 1.0/1.1” radio button</td>
</tr>
<tr>
<td>Query Template</td>
<td>The query template defines where the user’s query terms are sent to get an XML result set returned (Note the &amp;feed=rss parameter at the end of the example URL). For an OpenSearch location, the query template is a URL with the token {searchTerms} substituted where the query terms would go, and for a “Search Index on this Server” location it’s just the token {searchTerms}. There is an exception to using {searchTerms}, which is explained later in this chapter. This is a required field. IMPORTANT: The token {searchTerms} is case sensitive (all lowercase apart from the T).</td>
<td><a href="http://search.technet.microsoft.com/search/Feed.aspx?brand=technet&amp;query=%5C%7BsearchTerms%5C%7D&amp;feed=rss">http://search.technet.microsoft.com/search/Feed.aspx?brand=technet&amp;query=\{searchTerms\}&amp;feed=rss</a></td>
</tr>
<tr>
<td>“More Results” Link Template</td>
<td>The “More Results” link template is used in the Federation web parts on the Search Center results page to allow users to click through to see a full page of results rendered by the remote source, that is, Click Here to See More Results. It uses the {searchTerms} token in the same way as the query template. In this example, it would display an HTML results page from TechNet search.</td>
<td><a href="http://search.technet.microsoft.com/search/Default.aspx?brand=technet&amp;query=%5C%7BsearchTerms%5C%7D">http://search.technet.microsoft.com/search/Default.aspx?brand=technet&amp;query=\{searchTerms\}</a></td>
</tr>
<tr>
<td>Federated Search Results Display Metadata</td>
<td>The XSL field in the Federated Search Results Display Metadata section contains the XSL used to transform an XML result set into HTML that the Federated Search Results web part can display. If the “Use default formatting” box is checked, the default XSL is used, which will render RSS, ATOM, ATOM2, and RDF results in HTML. It is possible to replace or edit the default XSL, depending on your requirements.</td>
<td>Leave the “Use default formatting” check box checked.</td>
</tr>
</tbody>
</table>
Table 7-2 (Continued)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top Federated Results Display Metadata</td>
<td>The XSL field in the Top Federated Results Display Metadata section contains the XSL used to transform an XML results set into HTML that the Top Federated Results web part can display. If the “Use default formatting” box is checked, the default XSL is used, which will render RSS, ATOM, ATOM2, and RDF results in HTML. It is possible to replace or edit the default XSL according to your requirements.</td>
<td>Leave the “Use default formatting” check box checked.</td>
</tr>
<tr>
<td>Restrict Usage</td>
<td>The restrict usage section allows an administrator to control which Search Center sites can use this Federated Location. By leaving the default, No restriction, selected, this Federated Location can be used in any Search Center; by selecting “Use restriction,” an administrator can explicitly select in which Search Center sites the location can be used.</td>
<td>Leave the default “No restriction” selection.</td>
</tr>
<tr>
<td>Specify Credentials</td>
<td>There are several options in the Specify Credentials section for setting up authentication for Federated Locations. The settings in this section are used to authenticate to the location if required. They will be covered in more detail later in this chapter.</td>
<td>Leave the default setting of “Anonymous” selected.</td>
</tr>
</tbody>
</table>

All that’s left for you to do is click OK, and you’ve just created an OpenSearch 1.0/1.1 Federated Location for TechNet Search. The only thing left in terms of configuration is connecting the location you just created to one or more of the Federation web parts in your Search Center.

We’ll cover the Federation web parts and how to connect them to a Federated Location in detail in the next section, but first we need to go through what you’ve just done in a bit more detail. If you want some instant federation gratification, then hop down to the Federation Web Parts section and configure the Search Center web parts now, but just make sure that you come back to finish this section, because understanding how Federated Locations work is key to understanding how federation works, and we’re going to start with some deeper details on triggers.

**Triggers**

Triggers are set on a location, to control when a query will be sent to the location. It’s very likely that you will want to configure a location to be searched only when there are specific query terms or patterns in the user’s query. For example, you might want to display results from an Internet Search Engine location (Such as www.live.com) regardless of the user’s query, but you might want to query a parts database only when a specific phrase or pattern is matched in the user’s query terms.
Chapter 7: Extending Search with Federation

Using triggers in a smart way can make information much more discoverable, keep relevant results front and center on the results page, and keep results that are not relevant off the page.

There are three trigger settings available, and each is described in the following sections.

**Always**

Using the Always trigger setting will result in all of a user’s query terms being sent to the location every time a query is submitted. You should use this setting with care for a couple of reasons. First, you need to be sure that the location can handle the extra query load. Second, having too many locations set to Always will invariably end in a cluttered and confusing UI on the search results page.

**Prefix**

Prefix triggers only trigger the location when a specific term or set of terms is found at the start of the user query, and these triggers will send query terms found after the prefix terms to the location. For example, if the Prefix trigger is set to “Part Number”, and a user queries for “Part Number ABC123”, only the query terms “ABC123” will be sent to the location; the term “Part Number” will be dropped.

Prefix match terms are not case sensitive, and you can only use one set of match terms per location; the prefix terms will only be matched if they appear in the configured order. Considering the preceding example, if a user’s query terms are “Number Part ABC123”, rather than “Part Number 123ABC”, the location will not be triggered. Also, there’s no way to specify multiple prefix matches on the same location using this type of trigger.

**Pattern**

Pattern triggers are by far the most powerful and flexible triggers, and, of course, the most complex. They use a regular expression to match a pattern in the user’s query, and depending on how the expression is written, can be configured to send either all the query terms to the location or just the terms matched by the regular expression.

Explaining how regular expressions work is a book in itself. There are numerous online tutorials, so I’d recommend you search for “regular expressions,” using the Internet search engine of your choice, and spend some time getting acquainted. They’re an incredibly useful tool to have in your toolbox aside from Search, so the investment will be worthwhile. There are also several very useful visual tools for creating regular expressions. The one I’ve had most success with is Expresso from www.ultrapico.com; registration is free, and it has saved me countless hours in Notepad.

That being said, once you get the hang of how to create a regular expression you’ll probably find that you’ll hardly ever use prefix triggers because using a pattern trigger gives you a lot more control. For the rest of this section, I’ll walk through a few straightforward examples of pattern trigger regular expressions that can be reused and changed easily.

**Matching Numbers**

This is a simple example of how to configure a Pattern trigger to match a ZIP code (or any set of five numbers found one after another) anywhere within a set of query terms. The regular expression that follows will match a pattern of “numbers 0–9 exactly five times” and send all of the query terms to the location.

```regex
\d{5}
```
Chapter 7: Extending Search with Federation

For a sample query of “Bike resellers 98040”, the pattern trigger will match “98040” and send “Bike resellers 98040” to the location.

Matching Numbers and Letters
This example shows how to match three letters followed immediately by three numbers, which could be a client reference code, a support ticket number, or a part/stock number.

```
[a-zA-Z]{3}\d{3}
```

The preceding regular expression will match a sample query of “support case ABC123” or “stock number XYZ321” and will send all of the query terms to the location.

Creating a “Multiple Prefix Trigger”
This example shows how to create a Pattern trigger that acts like a Prefix trigger that can handle multiple prefixes. The regular expression that follows will match three prefixes: “Bike reseller”, “Car reseller”, and “Truck reseller”.

```
(?:Bike reseller|Car reseller|Truck reseller)
```

A sample query of “Bike reseller 98040”, “Car reseller Seattle”, or “Truck reseller North West” will match the trigger, and all of the query terms will be sent to the location.

Using Named Capture Groups to Send Only Specific Query Terms
Named capture groups are a regular expression feature that allows you to capture a set of specific terms from the input string (the user’s query terms), which, combined with configuring the pattern trigger in a specific way, allows you to send ONLY terms to the location that you want sent, rather than all of the user’s query terms. All of the previous examples in this section send all of the query terms a user enters to the location, when there’s a pattern match. Using capture groups, it’s possible to fine-tune exactly what part or parts of the user’s query terms are sent to the location, when the Pattern trigger is matched.

The following example shows how to capture “A word composed of the characters a–z or A–Z, followed by exactly one comma, and exactly one white space, followed by exactly two characters A–Z.” This means that it would match “Abcdef, XX”, “something, AB”, H, or probably more usefully, “Redmond, WA.”

```
(?<Address>[a-zA-Z]*,\s[a-zA-Z]{2})
```

For example, in this case, a user’s query of “hotels in Redmond, WA” would match the pattern set in the trigger on the terms “Redmond, WA”, and the named capture group “Address” will contain the terms “Redmond, WA”. This is highlighted in Figure 7-3.
The named capture group, which is the important bit for this example, is the \( (?<\text{Address}>\ldots) \) part. You can now use the capture group name as the token in the query template instead of using the token \{searchTerms\}. This means that only the text in the capture group is sent to the location, in this case “Redmond, WA”, rather than all the user’s query terms. The use of capture groups allows you to be much more precise with the query terms that are sent to the federated location.

The final step after setting up a Pattern trigger with a capture group is to modify the query template and “More Results” link template (if you’re using it) to match the capture group name you used in the trigger. For this example, we used the capture group name of “Address”, so it’s just a case of changing \{searchTerms\} to \{Address\} in the Query Template and “More Results” Link Template settings on the Edit Federated Location page for the location you’re configuring.

If you have an OpenSearch 1.0/1.1 location, the change to the query template would look something like this (and is shown in Figure 7-4):

Before:

http://server/results.aspx?q={searchTerms}&f=RSS

After:

http://server/results.aspx?q={Address}&f=RSS
And the change to the “More Results” link template would look something like this (and is also shown in Figure 7-4):

Before:

http://server/results.aspx?q={searchTerms}

After:

http://server/results.aspx?q={Address}

Figure 7-4: Using a named capture group in the Query Template and “More Results” Link Template

If you have a Search Index on This Server location, the change to the query template would look something like this;

Before:

{searchTerms}

After:

{Address}
Chapter 7: Extending Search with Federation

And the More Results link template would be the same as the Open Search example shown previously.

The token name {searchTerms} is reserved; you can’t use it as a named capture group name, and token names are case sensitive. The case used in the regular expression must match the case used in the query template.

Location Type

The Location Type option allows you to choose between OpenSearch 1.0/1.1 and Search Index on this Server. The selection you make controls how the user’s query terms are sent to the location. Setting this option to OpenSearch 1.0/1.1 will send the query to the location using the query template URL over HTTP or HTTPS, depending on the URL used in the query template, and it will allow you to choose and configure the authentication type (from the Restrictions and Credentials group further down the page), if the location requires authentication.

Setting this option to Search Index on this Server will send the query to the local index programmatically using the Search object model and using the credentials of the logged-on user. The authentication type cannot be configured when Search Index on this Server is selected (The Credentials UI is set to Default Authentication and can’t be changed).

Query Template

The Query Template section defines where queries are sent, and how they are sent. It will need to be configured differently, depending on the location type.

OpenSearch 1.0/1.1

For an OpenSearch location the query template must be a URL to a web page that returns XML, preferably as RSS, ATOM, ATOM2, or RDF, as there’s built-in XSL to render those schemas as HTML, so that the Federated Results web parts can display the results. If the URL returns XML in a different schema, everything will still work, but you’ll need to write custom XSL to render it as HTML.

When an OpenSearch 1.0/1.1 location is triggered, the federation framework will attempt to load the query template URL with the user’s query terms replacing the token {searchTerms} (or a custom token name configured through a capture group). There’s nothing special or complicated going on; you can literally cut and paste the query template URL into a browser, and whatever you get back is what the federation framework gets back. This also goes for authentication; if you try to load the query template URL in a browser and are prompted for credentials, you’ll need to configure the location authentication type to match the authentication type being requested, and enter a set of common credentials, or configure per user credentials (more on per user credentials in the next section).

A few query template example URLs are show here.

Example 1: The out-of-the-box query template URL for the Internet Search Results location:

http://search.live.com/results.aspx?q={searchTerms}&count={itemsPerPage}&first={startItem}&mkt={language}&format=rss&FORM=SHAREF
This query template URL would also work if it was just:


if you don’t need to control the things like the language preference for the results.

Example 2: The query template for a remote SharePoint Server or Search Server index:

http://sharepoint2007/SearchCenter/_layouts/srchrss.aspx?k={searchTerms}&start={startItem}

The URL is pointing at the Search Results RSS feed rather than the Search Results page.

Example 3: A query template URL that connected to an internal LOB application might look something like:

http://customerpoint/find/customersearch.aspx?customerName={searchTerms}&format=XML

Again, notice that the URL is specifying that the application at http://customerpoint return results as XML. If the application can return results as RSS, ATOM, ATOM2, or RDF, then the results will be rendered as HTML using the federated location’s default XSL. If it’s in a custom format, then the XSL configured for the location will need to be updated to reflect the schema the results are returned in.

Example 4: A query template URL for http://search.live.com that uses an advanced search keyword to return only results from the site “msdn.microsoft.com.”

http://search.live.com/results.aspx?q={searchTerms}+site:msdn.microsoft.com & mkt={language} & format=rss

This query template URL uses the site: advanced Live search keyword to restrict the results returned from the query to only those results from the site msdn.microsoft.com. Note that the site advanced search keyword is part of the q= query parameter and not an extra parameter, so the actual URL for a query of “SharePoint” would look like this:


This technique is actually very powerful and allows you to federate to locations that are publicly accessible but that don’t support OpenSearch, by using http://search.live.com as a proxy for the real site.

For example, if you replace site:msdn.microsoft.com in the previous URL with site:Wikipedia.com, you can get Wikipedia results through http://search.live.com even though Wikipedia doesn’t currently support OpenSearch.

If you want to take this a step further, it’s possible to create Live Search macros that can be used to create complex queries (see http://search.live.com/macros/default.aspx for more details). An example of a query template URL for a Live Search Macro is http://search.live.com/results.aspx?q={searchTerms}+macro:rriley.myfirstmacro & format=rss.
Chapter 7: Extending Search with Federation

It's also worth noting that other search engines in addition to www.live.com support OpenSearch, have rich advanced search keywords such as site, and can be used in the same way as the preceding examples.

**Required and Optional Query Template Tokens**

The only required query template token is \{searchTerms\} or a named capture group token substituted in its place.

There are several other optional query template tokens that can be used to insert values into the query template URL when the location is triggered at runtime. Table 7-3 shows the optional tokens and a description of their usage.

**Table 7-3**

<table>
<thead>
<tr>
<th>Token Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{startItem} or {startIndex}</td>
<td>This is used to specify the index of the result you want returned as the first result. This token can be used to page through federated results, but it is only really useful when implemented through code in a custom web part derived from one of the Federated Results web parts. If this token has no value when the location is triggered, it will default to 1.</td>
</tr>
<tr>
<td>{startPage}</td>
<td>Specifies the index of the result page you want returned first. This token can be used to page through federated results, but is only really useful when implemented through code in a custom web part derived from one of the Federated Results web parts. If this token has no value when the location is triggered, it will default to 1.</td>
</tr>
<tr>
<td>{itemsPerPage} or {count}</td>
<td>This is used to specify how many results the location returns. This token will be replaced by the value specified in the Federation web part property “Results Per Page” in the “Display Properties” group in a Federated Results web part property tool pane.</td>
</tr>
<tr>
<td>{language}</td>
<td>Allows locale information to be sent to the location. This token is replaced by the locale identifier set in the user’s browser (The locale identifier must conform to RFC3066). For example, if a user’s browser is set to “English (United Kingdom)” the {language} token will be replaced with “en-GB”. The location can use this information to customize the results it returns based on the language specified.</td>
</tr>
<tr>
<td>{inputEncoding}</td>
<td>Specifies the character encoding of the search request and is set to UTF-8 by default and cannot be changed.</td>
</tr>
<tr>
<td>{outputEncoding}</td>
<td>Specifies the character encoding that you want search results returned in and is set to UTF-8 by default and cannot be changed.</td>
</tr>
</tbody>
</table>
If we take the search.live.com example again:

http://search.live.com/results.aspx?q={searchTerms}&count={itemsPerPage}&first={startItem}&mkt={language}&format=rss&FORM=SHARE

and imagine a user has entered a query of “SharePoint”, the resulting query template URL that would be triggered would look something like the following URL, assuming that the user’s browser locale was set to English (United Kingdom).


**Search Index on this Server**

Federating to the local index might seem a little bit odd at first glance, but it’s actually a powerful and easy way to get a lot of control over how and when specific results are returned from the local index.

Configuring a location to use the Search Index on this Server location type allows you to show results from the local index in one or both of the Federated Results web parts. Using this functionality along with tightly scoped search scopes or property restrictions can yield some interesting and valuable new ways to help users find the content for which they were looking.

There are two configuration differences between an OpenSearch 1.0/1.1 location and a Search Index on this Server location.

1. Instead of using a URL for the query template, you can use either the token \{searchTerms\}, or, if you’re using a Pattern trigger with a named capture group, the token name you defined in the trigger. There’s no need to use a URL because the query is sent directly through the Search object model.

2. All of the credential’s settings in the Restrictions and Credentials Information configuration group are replaced with a read-only setting of Default Authentication. As Search Index on this Server locations use the local index, you don’t need to configure any additional credential information. They’ll pick up and use the credentials of the currently logged-on user and security trim the results appropriately.

Using just the token \( \text{searchTerms} \) as the query template won’t get you very far. You’ll have the Federated Search web part that you connect to the location duplicating the results shown in the Core Results web part, because all you will have achieved is to have sent the same query to the same index twice.

The key to using Search Index on this Server locations is the way you restrict the query in query template, so that the query returns a more targeted, smaller result set. Table 7-4 shows several examples of how to add restrictions to the query template, and they simply follow the SharePoint Server and Search Server keyword query syntax. Anything you can do with the keyword query syntax, you can do in the query template URL.
Chapter 7: Extending Search with Federation

Table 7-4

<table>
<thead>
<tr>
<th>Query Template value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{searchTerms}</td>
<td>Returns results from everything in the local index</td>
</tr>
<tr>
<td>{searchTerms} scope:Customers</td>
<td>Returns results from the Search scope &quot;Customers&quot;.</td>
</tr>
<tr>
<td>{searchTerms} site:<a href="http://intranet">http://intranet</a></td>
<td>Returns results based on the property filter site=<a href="http://intranet">http://intranet</a>.</td>
</tr>
<tr>
<td>{searchTerms} author:rriley</td>
<td>Returns results based on the property filter author=rriley.</td>
</tr>
<tr>
<td>{searchTerms} -author:rriley</td>
<td>Returns results based on the property exclusion of author=rrileyZ.</td>
</tr>
</tbody>
</table>

Thoughtful use of property filters and the inclusion and exclusion features of the keyword syntax in the query template allow you to restrict the volume and type of content the federated location returns.

Search Index on this Server locations, when used with a carefully configured trigger, can be a very powerful way to highlight or promote content that otherwise might be hidden, or not even visible, on a regular results page, selectively, from the local index to users.

More Results Link

The More Results link is the location to which you want users to be sent if they click the More Results link at the bottom of a set of federated results. The URL should point to an HTML version of the query template URL, or be left blank if there isn’t an HTML version of the results.

For example, the More Results link for http://search.live.com could be:

http://search.live.com/results.aspx?q={searchTerms}&mkt={language}

Notice that there is no format=RSS parameter because we want to get an HTML page back, and that the same tokens that are valid for the query template can be used. So, in the preceding example, clicking through the More Results link after a query of SharePoint with a locale of US English would load the following URL:


Display Information

The Display Information section in the location defines the XSL that will be used to transform the results returned from the location into HTML that will render in a readable way in the Federated Search web parts.

At the location level, individual XSL is configured for each of the Federated Results web parts. Having separate XSL definitions for both the Federated Search Results web part, and the Top Federated Results web part allows you to control the results display differently for each web part for the same location.
You might, for example, only want to return the top result in the Top Federated Results web part and display it in a specific way (maybe with a custom icon, background, or formatting), but if the same location were connected to a Federated Search Results web part, you would want it to render in a list with minimal formatting, no background, and no icon. The ability to specify XSL individually for each Federation web part allows you to control the rendering in this way at the location level.

If the “Use default formatting” check box is checked (it’s checked by default) the built-in default XSL will render RSS, ATOM, ATOM2, and RDF as HTML suitable for displaying in each Federated Results web part for OpenSearch locations. If the OpenSearch location you’re configuring supports returning results in one of those four formats, and you’re happy with the styling that the default XSL provides, you can leave the box checked and everything should work without further intervention.

It’s worth noting that a different default XSL is used for Search Index on this Server locations. Again, it’s different for each web part, and it transforms the SharePoint Server Results schema into HTML that’s suitable for displaying in the Federated Search web parts, and you’re also free to customize it if you need results from a Search Index on this Server location displayed in a custom format.

There are a couple of display items (the branding icon and the display title text) that are controlled by the XSL and are not configurable elsewhere) that you may want to have some control over, the branding icon and display text are shown in Figure 7-5.

The next sections cover how to customize both the branding icon and the display text by making a few simple changes to the default XSL.
Chapter 7: Extending Search with Federation

**Specifying a Branding Icon**

If there’s no branding icon specified in the XSL, it will default to showing no icon rather than a default icon. To specify a custom icon for the location to use, follow these steps:

1. In the Edit Location configuration page, scroll down to the Display Information section and uncheck the Use Default Formatting check box in either the Federated Search Results Display Metadata section or the Top Federated Results Display Metadata section (or both if that makes sense for what you want to do), and copy the contents of the XSL text box into Notepad or something similar (you can do this in the browser UI, but it’s just easier to copy and paste it).

2. Search the XSL for the following tag:

   `<xsl:param name="BrandingIcon">`<xsl:param>`

3. Insert the URL to your image into the XSL parameter tag, for example:

   `<xsl:param name="BrandingIcon">http://yourURL/yourIcon.gif</xsl:param>`

   The image should ideally be 16 x 16 pixels in size.

4. Copy and paste the updated XSL from Notepad back into the XSL text box in the location, scroll down to the bottom of the location, and click OK.

**Customizing the Title of a Federated Results Web Part**

By default, the text displayed in the title of the Federated Results web parts is taken from the RSS, ATOM, ATOM2, or RDF feed name. For example, the title shown in a Federated Search web part for a search for “SharePoint” using the built-in location for `http://search.live.com` would be “Live Search: SharePoint”.

Using the feed name isn’t always optimal, as you might want to put your own more descriptive text in the web part title. To edit the title text follow these steps:

1. In the Edit Location configuration page, scroll down to the Display Information section and uncheck the Use Default Formatting check box in the Federated Search Results Display Metadata section or the Top Federated Results Display Metadata section (or both if that makes sense), and copy the contents of the XSL text box into Notepad, or something similar (you can do this in the browser UI, but it’s just easier to copy and paste it).

2. Search for the following XSL tag:

   `<xsl:value-of select="$ChannelTitle"/>`

3. Replace `$ChannelTitle` with `$WebPartTitle` if you want to use the value set in the Title property in the Web Part tool pane.

   `<xsl:value-of select="$WebpartTitle"/>`

   or replace `$ChannelTitle` with "The text you want as the title here" if you want to set the title in the XSL for all web parts, rather than on a web-part-by-web-part basis as described earlier.

   `<xsl:value-of select=" The text you want as the title here "/>`. 
4. Copy and paste the updated XSL from Notepad back into the XSL text box in the location, scroll
down to the bottom of the location and click OK.

**Writing Custom XSL**

If the location you’re connecting to returns XML in a custom schema (or at least not in RSS, ATOM,
ATOM2, or RDF), you’ll have to write XSL, to transform the results into HTML that’s suitable for
displaying in one, or both, of the Federated Search web parts. Writing XSL is not a trivial task for most
people, and I’m not going to try and explain how to do it in this section!

What I will share is a handy snippet of XSL that will render the raw XML output from the location. This
is really useful for debugging or writing XSL, because you can see exactly what XML is being returned
from the location from within one, or both, of the Federated Search web parts. Follow these steps on the
location you want to configure:

1. In the Edit Location configuration page, scroll down to the Display Information section and
uncheck the Use Default Formatting check box in the Federated Search Results Display
Metadata section or the Top Federated Results Display Metadata section (or both if that makes
sense).

2. Click the ( . . . ) button next to the XSL text box, and type in the text below:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
     <xsl:template match="/">
       <xsl:copy-of select="*"/>
     </xsl:template>
   </xsl:stylesheet>
   ```

3. Click OK, scroll down to the bottom of the location, and click OK.

**Properties**

The properties set in the Properties text box define what result metadata is returned; the default settings
are different for an OpenSearch location and a Search Index on this Server location. For a Search Index
on this Server location, the default configuration is shown here:

```xml
<Columns>
  <Column Name="title"/>
  <Column Name="hithighlightedsummary"/>
  <Column Name="hithighlightedproperties"/>
  <Column Name="path"/>
  <Column Name="description"/>
  <Column Name="contentclass"/>
</Columns>
```
Chapter 7: Extending Search with Federation

The properties themselves are pretty self-descriptive and are used by the XSL to build the HTML result set. If you need to display additional metadata (for example, you might have a Federated Location that is scoped only to return Word documents, and you want the "Author" and "Size" properties displayed within your results), then you’ll need to add additional <Column Name="ManagedPropertyNameHere"> elements for each of the properties you want to add. You can add any of the managed properties configured on your server (found through the Metadata Properties link on the Administration dashboard on the left side).

Just adding <Column Name=""> elements won’t make the properties magically appear in the results shown in the Federated Results web parts. After you’ve added your custom properties, the final step is to modify the XSL, so that it understands how to render them.

Properties are ignored for OpenSearch locations, so even though the defaults are set to look relevant for an OpenSearch location, as shown here, they’re not actually used when the location is queried. Adding extra <Column Name=""> elements to an OpenSearch location will have no effect on the properties returned from the location.

<Columns>
  <Column Name="title"/>
  <Column Name="link"/>
  <Column Name="description"/>
</Columns>

To have extra properties returned for an OpenSearch location, you’ll need to change the source to return additional or different properties, and update the XSL accordingly. This implies that you own the source of the data and can configure it. If you can update the source, or if additional properties are being returned and the default XSL isn’t rendering them, you can just update the XSL; you don’t need to update the properties settings.

Sample Data

The Sample Data sections in the Display Information section are used by SharePoint Designer to render sample content in the Federated Search web parts when the page is edited in its design interface.

You’re free to change this sample content to anything that makes sense for what you’re doing, although in practice, it’s rarely worth the effort, since it’s only seen when the page is in design mode within SharePoint Designer, and you’re not likely to be using SharePoint Designer to create a custom XSL to test against it.

Sample data comes in handy if you’re writing custom XSL in Visual Studio, or something similar, and want something to test against your XSL. Cut and paste the contents of one of the sample data sections into Notepad and save it as .xml, and you have yourself a small, but perfectly functional, test result for RSS, ATOM, ATOM2, and RDF. If you switch the location type to Search Index on this Server and do the same thing, you’ll get a sample result in the SharePoint Search results schema.
Chapter 7: Extending Search with Federation

**Restrictions**

The last group on the Edit Federated Location page, the Restrictions and Credentials Information group, includes the ability to configure the Search Center sites within your SharePoint Server farm or Search Server deployment that are allowed to connect Federated Results web parts to this location.

The settings for this configuration option are described in Table 7-5.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>No restriction: All sites can use this location.</td>
<td>This is the default setting and will allow this location to be connected to a Federated Search web part in any Search Center site within the SharePoint Server Farm or Search Server deployment.</td>
</tr>
<tr>
<td>Use restriction: Only allowed sites can use this location.</td>
<td>Configuring the location with this setting allows you to set, explicitly, a list of Search Center sites that can be used with this location. To connect a Federated Search web part to this location with this configuration setting enabled, the Search Center site you want to use this location in must be listed in the Allowed Sites: text box. This location will not be visible to any Search Center sites that are not listed in the Allowed Sites: text box. If you need to add multiple sites to the list, they must be separated with semicolons.</td>
</tr>
</tbody>
</table>

Using the restriction setting, it’s possible to manage whether a specific location shows up in the Federated Search web parts as an available location. This is great way to control which sites in your farm can use the location on a Search Center–by–Search Center basis.

As an example, if you have a site collection to which you don’t have admin rights, that’s used for an HR portal, which has its own Search Center, you might want to configure several HR related Federated Locations for that Search Center, but probably don’t want the admin responsible for the Sales and Services portal, which is in a different site collection, to be able to add them to their Search Center site. By adding the URL to the HR Portal Search Center to the “Allowed Sites:” list, you can lock the location down, so that only that site can use it.

**Credentials**

There are many locations you can federate to anonymously, such as publicly accessible search engines like [http://search.live.com](http://search.live.com). In a corporate environment, however, things are usually not simple, and the security and confidentiality of results are key requirements and a very real business concern.
Chapter 7: Extending Search with Federation

The credentials section in the Restrictions and Credentials Information group allows you to configure several different types of authentication in three different ways. In the next three sections, we’ll cover the types of credentials and the different ways that they can be passed by locations. It’s worth noting now that these options only show up if the location type is set to OpenSearch. You can ignore the next three sections if you’re configuring a Search Index on this Server location, as those locations will use the currently logged-on user’s credentials (i.e., the user that’s running the query), and it’s not possible to configure the location to use anything else.

**Anonymous**

Anonymous is pretty straightforward and works exactly the same way as you do, when you’re browsing the web. If the location is triggered, the query template URL is loaded, and no credentials are used. If the location returns an access denied, or some other error, then no results will be returned, and the Federated Search web parts connected to the location and the web parts will not render themselves.

**Common**

Common credentials are also pretty straightforward and allow you to configure a common set of credentials that will be used every time the location is triggered. An example of a location that might require common credentials to be configured is a subscription news or financial search service to which your company subscribes, and that is available to all employees, but uses a single shared username and password for everyone.

The types of authentication supported for “Common” locations are listed below:

- Basic Authentication. (You specify the credentials.)
- Digest Authentication. (You specify the credentials.)
- NTLM. (This uses Application Pool Identity.)
- NTLM. (you specify credentials)
- Form Authentication. (Fill a form in on a web page and press the “Submit” button.)
- Cookie Authentication. (Navigate to a web site, and it stores the cookie returned after logging on to the site.)

Once you’ve selected and configured the relevant credentials for the location, every time the location is triggered, it will try to load the query template URL and send the credentials you set, in the format you selected. If the location returns an access denied, or some other error, then no results will be returned, and the Federated Search web parts connected to the location and the web parts will not be rendered.

**User**

This is where it gets a little complicated. Using “User” credentials allows the location to pass the currently logged-on users’ credentials to the location, when the location is triggered. This is very useful for some external subscription sites, as well many internal “enterprise locations,” such as archive servers, email servers, and databases.
Chapter 7: Extending Search with Federation

Sending the credentials of the user that is running the query to the location allows it to do things like security trim the results, or maybe even return an access denied if appropriate. However, there are some implementation differences between using Kerberos and the other authentication types available in this configuration setting.

The types of authentication supported for User locations are:

- **Kerberos** — Logged-on users’ credentials are passed automatically.
- **Basic Authentication** — Logged-on users’ credentials are not passed automatically.
- **Digest Authentication** — Logged-on users’ credentials are not passed automatically.
- **NTLM** — Logged-on users’ credentials are not passed automatically.
- **Form Authentication** — Logged-on users’ credentials are not passed automatically.
- **Cookie Authentication** — Logged-on users’ credentials are not passed automatically.

As you can see from the list above, the only User authentication type that works out-of-the-box is Kerberos. If your location supports Kerberos, then you’re in luck; it’s just a case of selecting the radio button, clicking OK at the bottom of the screen, and you’re done.

There is a slight complication if you want to use anything other than Kerberos from the preceding list. You’re free to select any of the other authentication types in the UI. If you select Basic Authentication, for example, you’ll still be able to click OK and the location will be saved without an error, but when you connect the location to a Federated Search web part, it will never return any results. You’ll probably then be left wondering what happened and spend a few hours trying to figure out what went wrong.

The reason that no results are returned is that the Federated Search web parts have no way of gathering a user’s credentials (apart from Kerberos) and automatically sending them on. If you need to use Basic Authentication (or any other User setting apart from Kerberos), you’ll need to do a few extra steps that require some Visual Studio work, if you’re using Search Server; or some work configuring the SharePoint Server single sign-on service, if you’re using SharePoint Server.

**Configuring Search Server for “User” Authentication**

The steps that follow outline the process you need to follow to configure Search Server to support User authentication for anything other than Kerberos.

- Create a Federated Results web part that includes a UI to let users enter their credentials. Both of the Federated Results web parts can be overridden to achieve this.
- Write code to submit the credentials when required.
- Optionally, write code to store the user’s credentials in a cookie or an SSO (single sign-on) service.
Chapter 7: Extending Search with Federation

There’s a very thorough article on msdn.microsoft.com that walks through each of these steps, including details on how to deploy the custom web part after you’ve finished in Visual Studio. Searching for “Custom Federated Search Web Part” at site:msdn.microsoft.com with the Internet search engine of your choice should help you find the article and sample code.

Configuring SharePoint Server for User Authentication

If you’re using SharePoint Server (with Enterprise CAL) the process is a little easier, provided that you’ve set up and configured the single-sign on service.

In addition to having the option of creating a custom web part to capture users’ credentials, you can select to use a configured SSO application to manage per user credentials automatically. Once your SSO application has been configured (see the SharePoint Server Administration and Deployment guides for details), you simply need to check the Use SSO check box for the per user authentication type you want to configure for the location, and then select your SSO application from the list as shown in Figure 7-6.

Security Trimming Federated Results

Security trimming is the process of removing any search results to which the user doesn’t have access rights before the results are rendered in their browser. This process prevents users from seeing results, and, in particular, document snippets that might be considered a security or confidentiality risk.

There are two ways to achieve a security-trimmed result set with Search Server and SharePoint Server for regular results, that is, results from the local index rendered in a Core Results web part (ignoring anything to do with federation for a moment).
The first is automatic. When several types of content sources are indexed (including File Shares and SharePoint data), the indexer will index and store the security information as well as the content of the document, and any results from those content sources can be automatically trimmed, if the user doesn’t have access to them before they’re returned to the user.

The second method is not automatic and requires the use of a Custom Security Trimmer, which is custom code to intercept a result set and insert your own logic to decide what gets trimmed. You could create code to step through each result and test in real time to see whether the user has access to the resource, stripping any out to which they don’t have access, before displaying the results.

Security trimming results from federated locations is slightly different.

If the location is a Search Index on this Server location, then the automatic security trimming happens because you’re using the Search object model to get results, and the user’s credentials are used whether you like it or not — which makes sense because from a security perspective it behaves in the same way that the Core Results web part does. Any Custom Security Trimmers you may have configured are also respected, so from a security perspective, results from a Search Index on this Server location behave exactly the same way as results displayed in a Core Results web part.

There is a difference, however, if the location is an OpenSearch 1.0/1.1 location, because there’s nothing you can do from a Search Server or SharePoint Server perspective in terms of security trimming results. Whatever results are returned from the location are rendered in the web part. There is a limited amount of logic that you could write into the location’s XSL to do something like hide or ignore certain results, but using XSL to achieve this would be more “security through obscurity” than true security.

The only way to ensure that results are correctly security trimmed for an OpenSearch 1.0/1.1 location is to do it at the source, and that’s what the User credentials’ setting on the location is about.

The Federated Results web parts are not designed to “understand” the results they get back, or to do anything other than render them, so the results need to be “pre-security-trimmed” by the location before it sends them back to Search Server or SharePoint Server. Sending the user’s credentials to the location, when the location is triggered, gives it the information it needs to trim the results set, so it only returns results that that specific user has access to. Again, this does make sense; you’re federating, not indexing, and one of the main reasons do so is that you want to send a query and get results without having the overhead of maintaining an index and other data, such as access control lists for that information. We want the OpenSearch 1.0/1.1 location to do the clever stuff, rather than us.

It’s highly likely that many locations you would want to federate to and get security-trimmed results from, such as an email archiving server, will already have a security model in place for the search engine that it uses. The location is probably configured to return trimmed results by default. If it isn’t, then there’s not much point worrying about the results you’ll get back in Search Server or SharePoint Server because you’ve already got a bigger problem that needs to be fixed.
Chapter 7: Extending Search with Federation

Federation Web Parts

After you’ve finished creating and configuring a Federated Location, the last thing you’ll need to do is to connect it to a Federated Search web part, so that users can see results from the location displayed in a Search Center.

There are two Federated Search Results web parts: the Federated Search Results web part and the Top Federated Results web part. Both web parts perform similar tasks in that they display results returned from a Federated Location, but they’re slightly different in the way they display results, and how they’re configured. Figure 7-7 shows both web parts displayed in a search center results page, the Top Federated Results web part shows a result from Wikipedia, and the Federated Search Results web part shows results from www.live.com.

Figure 7-7: Search Center Results page showing both Federated Results web parts
The rest of this chapter covers how to configure and customize both of the Federated Search web parts. They share a lot of similar configuration settings, so we’ll cover the Federated Search Results web part first (it’s likely to be the one you’ll use most frequently), and then go through the differences for the Top Federated Results web part.

**Federated Search Results Web Part**

Federated Results web parts can be added to a Search Center site results page just like any other web part. Navigate to the results page by searching for something (it doesn’t matter what it is), put the page into Edit Mode from the Site Actions menu, click on the Add Web Part link for the web part zone to which you want to add the web part, then scroll through the list of web parts and choose Federated Search Results Web Part, click OK (and the web part magically appears on the page in the web part zone you picked). Figures 7-8 and 7-9 show these steps.

![Figure 7-8: Search Center results page in edit mode](image-url)
Once you have a Federated Search Results web part on your results page (if you’re running Search Server, there will be two of them already on the page), there are several configuration options that can be set to control its behavior. Figure 7-10 shows the Search Center site results page in edit mode, with the Federated Search Results web part tool pane open, showing some of the configuration options for one of the Federated Search Results web parts.
Chapter 7: Extending Search with Federation

Federated Search Results Web Part Properties

The next few sections walk through some of the more common configuration settings, and explain what they do, and when you might want to use them.

**Location**

Location is the most important setting on the web part. This setting connects the web part to a single location from which you want to display results. Using the drop-down list, you can select the location by choosing the relevant display name, and if you have several locations configured with similar names, you can also check the description field to ensure that you’ve picked the right one.

If, after configuring a new Federated Results web part, you see no results displayed, it’s possible that one or more of three things could have happened.

1. Your location isn’t configured correctly. Maybe the query template was wrong, or the security settings were misconfigured. Go back to the Edit Location Definition page and try cutting and pasting the query template URL into a browser to see what happens. That should be the starting point for troubleshooting.
Chapter 7: Extending Search with Federation

2. The location is configured correctly but isn’t returning any results for the query you submitted. If the location returns zero results, the Federated Search web parts won’t be rendered (that is, they won’t be rendered at all), so don’t expect to see a title bar for your location and empty space below. If the location returns nothing, then the web part effectively hides itself to save screen real estate. Try a different query that you know will get results back from the location.

3. The location is configured correctly but isn’t returning any results for the query you submitted, even though you know you should be getting results. If this is the case, the chances are that you’ve configured a trigger for the location, and the query you’re testing with is missing the trigger prefix or pattern. Try a different query that includes the trigger prefix or pattern.

**Results per Page**

Results per page sets the number of results displayed in the web part. If the location supports a parameter that can do something with the `{itemsPerPage}` or `{count}` tokens, then those tokens will be replaced by the setting in this configuration option, and only that number of results will be requested and then displayed. If the location doesn’t support a parameter that can set the number of results returned, then this option will take the top X from whatever the default result set size is for the location and display them in the web part, with X being the number you specify in the Web Part property.

**Use Location Visualization**

If checked, Use Location Visualization configures the web part to use the XSL set at the location level to render results. If this box is unchecked, you get the ability to replace the location XSL on an individual web part by clicking the XSL Editor and pasting in a new XSL document. I don’t recommend using the Editor to write your XSL because it’s not much more than a text box (in fact I think it is a text box).

**Retrieve Results Asynchronously**

By default, a Federated Search Results web part will be rendered asynchronously. This prevents it from holding up the rest of the results page that is loading, if it’s waiting to get results from a slow location. If you’ve configured multiple Federated Search Results web parts on the page, and they are all configured to be asynchronous, they will be rendered in the order they get results. This could make the page jump around a little, because the web part location on a page doesn’t change based on the order in which the web parts get results; rather, it’s set by the order in which they are placed within the web part zones on the page.

If you uncheck the Retrieve Results Asynchronously check box for a Federated Search Results web part, the results page will not be rendered until that web part has either gotten results or timed out (after 90 seconds). If you have locations configured that can be slow to return results, we recommend that you leave this configuration setting set to the default, which is asynchronous.

**Results Query Options**

All of the settings found in the Results Query Options group are only relevant for Search Index on this Server locations and are grayed out for OpenSearch 1.0/1.1 locations. They perform exactly the same functions as the options you’ll find on a Core Results web part.
Chapter 7: Extending Search with Federation

Show More Results Link

The Show More Results link enables you to switch on or off the More Results . . . link at the bottom of the results in the web parts. It also allows you to customize the text you want displayed for the link.

Appearance Settings

Settings in the Appearance group control the configuration of the web part chrome, title, and size. These settings are exactly the same as pretty much every other out-of-the-box web part, and you can play around with them to get the Federated Search web parts to look exactly the way that you want them to for your Search Center. You’re likely to be poking around in this section if you change the number of results, the number of characters in the summary, or the number of characters in the URL to anything other than their default settings, because you’ll probably end up with some ugly scroll bars on the web part that you’ll want to get rid of.

Caching Location Settings

Caching isn’t a setting that appears in the Web Part properties, and you can’t configure it, but it’s worth knowing that it’s there. All of the settings that Search Server or SharePoint Server needs to execute a query to a location are cached, not the query or any results, but the configuration elements such as the query template and XSL. This is done to reduce the increased load the server would otherwise have when looking up these details every time the location was triggered, which, if a trigger were set to Always, would be every time a user searched for something.

In normal day-to-day activity, you’re unlikely to notice that this is going on, or even care that it’s happening, but if you’re trying to debug or troubleshoot a problem, then you absolutely need to know that it’s happening, or else you can find yourself running in circles, trying to figure out why that update to the XSL you just made isn’t being reflected in your web parts.

You can’t change or turn off the caching settings, and they’re set to 60 seconds, so the worst-case scenario is that you update the query template or XSL on a location, and it takes 60 seconds for the change to appear in the web parts. An IIS reset will force an update, but it’s probably quicker just to wait — but at least you’ll know why you’re waiting.

Top Federated Results Web Part

The Top Federated Results web part is pretty much the same as the Federated Results web part in terms of configuration and operation apart from two key differences, which are explained below.

Multiple Locations

The Top Federated Results web part can be connected to up to 10 different federated locations at once, compared to the 1:1 mapping for a Federated Search web part. If you take a look at Locations property under the Web Part properties of a Top Federated Results web part, you’ll notice that there are two location drop-down boxes, and Add and Remove buttons to add more locations or remove existing ones.

This web part is designed to display the top result from the first location in the list of configured locations that returns a result set in a Top Answer format. When a user submits a query, each location that’s configured is queried in the order that it appears in the Locations Web Part property, until one of them returns a set of results. From the results that are returned, the top X (only the first by default) is displayed.
Chapter 7: Extending Search with Federation

If none of the locations returns any results, then the web part isn’t rendered. It’s also worth noting that the order in which you configure the locations in the list is important. You should aim to go from your most authoritative location to the least authoritative location, working from top to bottom.

Using the Top Federated Results web part with multiple locations that have well-crafted pattern match triggers is a great way to push very relevant content out to users without having to have lots of Federated Search web parts configured on the page, or to have several of them return results at once, complicating the page layout.

**Retrieve Results Asynchronously**

The other major difference between the Top Federated Results web part and the Federated Search Results web part is that the Top Federated Results web part is set to be synchronous by default and cannot be set to asynchronous.

Because the web part usually lives at the top of the results page, above the Core Results web part, in the event that the results from the Top Federated Results web part are rendered after the Core Results web part has been rendered (which is likely, given that it might trip through several locations before hitting a result), the web part is forced to operate synchronously to prevent it from knocking the page content down, including results that users were probably already reading.

To avoid a potentially negative user experience, it will always be rendered synchronously with the other synchronous web parts on the page (including the Core Results web part). What this means in real life is that if you have several slow locations configured in a Top Federated Results web part that maybe take 5 seconds to return results, then your results page will take 5+ seconds before it renders. Essentially, your page load time is as fast as the slowest location configured (and triggered) in your Top Federated Results web part. This is something you’ll want to keep an eye on, as it could also negatively impact your user’s Search experience.

**Summary**

Hopefully, this chapter has given you some insight into the power and flexibility of the federation features in SharePoint Server and Search Server. Federation is a great tool to have in your tool belt and when configured thoughtfully can really make a difference to your user’s Search experience.
Securing Your Search Results

Security is an important part of any Enterprise Search system. Since most indexing engines run as super users in your environment, giving them access to all content, you want to make sure that the search engine is returning only results that the user making the query would have permissions to, rather than all results. For example, you do not want the Excel spreadsheet with everyone’s Social Security number and salary information being returned to unauthorized users. Many companies, when they implement SharePoint for Search, suddenly discover all the sites that users are not correctly locking down from a security standpoint. SharePoint is not only a great search engine, but it is also a great way to uncover bad security in your environment (wink, wink)!

In this chapter, you will take a look at four key areas. First, you will look at the default security architecture for SharePoint, and how the crawler and query engine work together to provide security for some datasources. Second, you will look at forms-based authentication, and writing your own security trimmer for the times when the OOB security trimmer does not meet your needs. Third, you will look at the security architecture provided by the Business Data Catalog (BDC). Finally, you will take a quick look at securing the search service in SharePoint.

Security Architecture in SharePoint

Security is a core part of SharePoint search. SharePoint supports security features such as item-level access control lists (ACLs), security trimming (which removes items from the user interface or result sets before displaying them to the user), and the ability to create custom roles with custom security settings. Therefore, you cannot turn off the security trimming that is built into the product, since any query that touches a datasource that supports security trimming will be trimmed automatically. The only way to access untrimmed results is to run in the context of an account that has access to all content, so that trimming is minimal. Normally, you can do this via logging in as the specified account, or via code by calling the SPSecurity method RunWithElevatedPrivileges and calling the Search object model.
Chapter 8: Securing Your Search Results

SharePoint gets security information from a datasource at crawl and index times. When the crawler hits a datasource that supports returning ACLs, such as file shares, SharePoint sites, or Exchange or Lotus Notes datasources, the crawler asks the system for the security descriptor related to the document using the GetSecurityDescriptor method. Depending on the protocol handler used, the process may be a little different. For example, Lotus Notes uses Lotus Notes groups and users. The Notes protocol handler maps Notes user ACLs to Windows ACLs so that SharePoint can trim Notes results. Unfortunately, SharePoint does not support mapping Notes groups to Windows groups. One thing to note is that the BDC does support security trimming but only at the entity level.

The ACLs that SharePoint gets back are stored inside SQL Server. If you ever want to see what is actually stored, just go into your search database and look at the MSSessionDocSdids table. Please note that you should not change anything in this table, and only view this table on a test server, never a production server. You will find that table contains the document ID, the type of security, whether it's NTLM (0) or custom through pluggable authentication (1), the security descriptor ID, and whether a pluggable security trimmer is associated with the document (0 – no, 1 – yes). There is a special value of 0 for the Sdid column, which means that everyone can see the content.

You could also look at the MSSecurityDescriptors table, which maps the security descriptor id to the security descriptors in your system. The sdid is the ID you will find in the document security descriptor table discussed earlier. The type specifies whether it is an Windows NT/Lan Manager security (NTLM) or uses pluggable authentication. Finally, the security descriptor itself is stored in binary format in the SD column.

One thing you will want to remember is that security changes are only picked up on crawls, so if you change security frequently on a data source, you will want to crawl that data source frequently to pick up any security changes. Storing the ACLs you get while crawling and indexing is called crawl time ACLs and trimming, SharePoint also supports query time trimming, where, rather than comparing ACLs between the stored database ACL and the user running the query, SharePoint executes a real-time check between the search result items’ permissions and the user executing the query. The query time method gets you the immediate permissions for the user, since the crawl time ACLs are only as fresh as the last time the data source was crawled. You pay for the freshness with performance, since checking permissions at query time can be slower than using the stored ACLs.

One data source that does not support security descriptors is web-based content. Web sites normally do not pass back ACLs on their web pages, and often you cannot call the web application to get this information. Therefore, SharePoint cannot store the ACL for the web page in the index. For web security trimming, you will have to use the query time security trimmer and trim out the results on your own from your web-based sources. SharePoint has some new capabilities that may make this easier, especially the support for forms-based authentication and cookie security trimming, but you may find that you need more granular trimming to meet your needs.

When a query is executed, SharePoint determines the raw set of documents that match the query terms. Then, the results are passed through security trimming. SharePoint removes documents from the results if the person performing the query does not have permissions for those documents. If a security trimmer is registered, the trimmer takes over and trims the results even further. The final results are returned to the end user.
Chapter 8: Securing Your Search Results

Best Bets
One thing to be aware of is that Best Bets for search results are not security trimmed because SharePoint does not place ACLs on best bets, and best bets are shown in their own web part, separated from the search results. Therefore, you could be exposing sensitive information by accident through the best bets technology. The easiest way to secure your environment completely is to remove the Best Bets web part from your search results page. The Best Bets web part is the high-confidence web part.

If you are not that security minded and do not want to lose the best bets functionality, just remember that best bets are not trimmed when you create them. Imagine that all users will see what you are creating as a best bet and use the correct judgment whether to create a best bet.

Controlling Indexing to Secure Content
SharePoint allows you to control which content is crawled by the engine at a granular level. This is to provide you with the ability not to crawl certain content. This can be used to secure your crawls and search results. For example, if you have a site that has a secured list, and you have a page on that site that displays that list through a web part, you will want to disable crawling of that page. The reason is that the crawler only checks permissions at the page level and will crawl the content on that page and index the content in the secured list. A person searching may not have permissions for the list but may have permissions for the page, which then would return data in the search results that the user should not be allowed to see.

What about IRM-Protected Documents?
Information Rights Management allows you to secure your documents with policies, such as stopping the ability to print, save, or forward the document. Normally, search engines cannot work with IRM documents because of the security applied to them; IRM documents are encrypted. Since documents, themselves can be secured using IRM, the search engine has to know how to work with IRM-enabled documents, and also how to make sure that relevant results are returned, regardless of their IRM statuses. SharePoint search has built-in integration with IRM. In order for the crawler to be able to work with IRM content, the content must be IRM enabled by SharePoint after being put into a SharePoint library. For example, if you IRM enable a document locally and then upload it to SharePoint, SharePoint will not be able to crawl the document and return it in the results. SharePoint, not the end user, must apply the IRM policy.

Therefore, you will want to upload your document to a SharePoint library that has the IRM policies enabled in SharePoint. SharePoint will then apply the IRM policies at request time for the document, rather than at crawl time. When a user downloads a document or opens it, SharePoint applies the IRM policy on demand. This architecture allows searching of secured content.

Custom Security Trimmers
There are times when you will want to implement your own security trimming, such as crawling back-end systems using the BDC or web content, which do not support storing ACLs at index time, and where using Forms Based Authentication (FBA) or cookie authentication may not meet your needs. SharePoint provides an API to implement your own security trimming through the ISecurityTrimmer API in the Microsoft.Office.Server.Search.Query namespace. One thing to realize is that the security
Chapter 8: Securing Your Search Results

trimming API is only provided on the Search Server 2008 and Office SharePoint Server products and not on the Windows SharePoint Services (WSS) product. The API allows you to trim the results even further than what SharePoint trims. It does not allow you to add new things to the search results.

Implementing a Custom Security Trimmer

A custom security trimmer is a .NET assembly that you create and register on your SharePoint server. You register your trimmer by mapping it to crawl rules, so that the trimmer is implemented on the particular content that the crawl rule you attach it to returns. To implement the ISecurityTrimmer interface, you need to implement two methods — Initialize and CheckAccess. The Initialize method initializes your security trimmer and is called whenever the trimmer is loaded into the worker process, and whenever the worker process is recycled. This method passes two parameters. The first is a name value collection using the System.Collections.Specialized.NameValueCollection. This collection contains the values you specified when you register your custom security trimmer.

The second parameter is a SearchContext object, which is your entry point to the search service for SharePoint. From this object, you can access the administration methods and properties for your search service.

The second method is the CheckAccess method. This method is executed at least once per instance that a search result is returned for a particular crawl rule for which the security trimmer is registered. This method can be called multiple times, depending on the number of results returned from the query. The query processor dynamically adjusts the number of results, which affects the number of calls to the CheckAccess method. There are two parameters passed to this method. The first is System.Collections.Generic.IList object, which contains string values for all the URLs for the items that matched the query from the index. The second is a System.Collections.Generic.IDictionary object, which you can use to track information across multiple CheckAccess method calls. This method returns a System.Collection.BitArray object, which contains true or false values that correspond to the item URLs in the first parameter. You set the value to true to include the item in the results, and false to exclude the item.

You can retrieve the identity of the user who submitted the query to help you decide whether the result should be returned or excluded. The way you retrieve the identity will be determined by the type of authentication used. For example, if you used Windows Authentication, you will use the WindowsIdentity.GetCurrent().Name method, which will return a string with the username of the person who submitted the query. If you are using forms-based or custom authentication, you will use the HttpContext.Current.User.Identity.Name property, which returns the name of the current user making the search query.

You will deploy the assembly you create to the GAC on all the machines that are query servers. So, if you have scaled out your query servers to three servers, you will need to place your assembly into all three GACs.

One easy way to deploy your solution is to use SharePoint solutions or, as they are known, WSP files. By creating WSP files, you can have SharePoint deploy your security trimmer DLL to the GAC and register it.
Chapter 8: Securing Your Search Results

Registering the Custom Security Trimmer

You associate your security trimmer with an SSP and a crawl rule. Trimmers can have a one-to-many relationship with SSPs and crawl rules, but you will need separate registrations for each of those relationships. For example, if you have three SSPs for which you want to use the same trimmer, you will need to create three registrations associating that trimmer with those SSPs and crawl rules. STSADM has a command to register your trimmer called `registersecuritytrimmer`. This command takes the four required parameters and one optional parameter, described in Table 8-1:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssp</td>
<td>Yes</td>
<td>The name of the SSP to register the trimmer with.</td>
</tr>
<tr>
<td>id</td>
<td>Yes</td>
<td>A unique identifier for your trimmer. If two trimmers are registered with the same name, the last trimmer to register will win.</td>
</tr>
<tr>
<td>typeName</td>
<td>Yes</td>
<td>The strong name of the assembly for the trimmer.</td>
</tr>
<tr>
<td>rulepath</td>
<td>Yes</td>
<td>The crawl rule the trimmer should be associated with.</td>
</tr>
<tr>
<td>configprops</td>
<td>No</td>
<td>A name value pair that specifies custom configuration information you want to pass to the trimmer from the registration. The format should be Name1<del>Value1</del>Name2<del>Value2</del>etc.</td>
</tr>
</tbody>
</table>

Make sure to reset IIS after completing your registration. You will also need to perform a full crawl on your content source after registering the trimmer because SharePoint needs to set properties, which you saw earlier, in the index to identify that the content has an associated custom security trimmer. You will see an example of the command line used in the sample you build later in this chapter.

Performance Considerations

On the registered SSPs and crawl rules, your trimmer will be called for every query that returns results. This means that you need to write well-performing code, since poor coding could considerably slow down your SharePoint searches. SharePoint will continuously call your trimmer on large result sets unless you tell it to stop. This means that if you get back a million results and the user only has permissions to result number one million, your trimmer will be called for every result set before hitting that millionth result. There are a couple of best practices you can follow to try to reduce the impact of your code on the performance of your SharePoint searches.

First, you will want to implement a limit on the number of security checks that your code will perform for a single query. If you are performing too many security checks, you will want SharePoint to return a message to the user telling that person to refine his or her query to get a smaller result set. To do this, you can keep a counter on the number of security checks your code has performed, and after some threshold (which you can make customizable by your IT staff by allowing them to pass it in the
registration in the name/value pair) you can throw the PluggableAccessCheckException exception. This exception takes in its constructor a string for which you can specify the text to return to the user, rather than the results of the query. This text should tell the user to refine the query to limit the number of results.

Second, you will want to think about caching security permissions, either locally on the query servers, where your trimmer is running, or remotely in another data store. This caching allows you to mimic the built-in crawl time ACL harvesting of the crawler, which will make your security trimmer faster. Rather than having to call the content source to check ACLs for each item every time your trimmer is called, you can instead just look it up in your cache. There are some downsides, however; you need to write the code to get the content’s ACLs from the source system to put into the cache, and your cache is not real time. Therefore, if you are using trimming for a highly ACL dynamic environment, then you will want to take the performance hit of real-time access checks. It has been my experience that many environments do not change permissions rapidly, and a 12- or 24- hour refresh is real time enough for most scenarios.

**Bringing It All Together: Building a Custom Security Trimmer**

Enough chit-chat about building a custom trimmer; let’s take a look at coding one up. The first thing you need to do is create your security trimmer in Visual Studio (VS). The scenario that you will look at here is writing a trimmer that trims out results from a forms-based authentication site, based on a user role. There may be sections of a site that you want certain users to have access to, and others that you don’t, based on a role. The sample uses the built-in ASP.NET SQL Server user and roles database called aspnetdb. The sample looks at the roles passed into the registration for the security trimmer that will identify the users who have access to the content. If the user performing the search is not a member of any of the roles, the results will be trimmed out. Since the trimmer can be registered with multiple crawl rules, you can control the security by making the crawl rules correspond to the areas you want to secure. SharePoint makes setting up FBA crawling easy, but it still does not support out-of-the-box security trimming.

**Getting Started**

The first step in writing your security trimmer is to fire up Visual Studio. You will want to write a Class Library. Make sure to add a reference to the Microsoft.Office.SharePoint.Search assembly. You will also want to add a reference to System.Web if you are using forms-based authentication. This will allow you to use the HttpContext class from which you will get the username of the current FBA user.

As for your Using directives, you will want to make sure that you add directives to SharePoint search, collections, and other namespaces as shown in the following code:

```csharp
using System;
using System.Collections.Generic;
using System.Text;
//Windows Authentication
using System.Security.Principal;
```
Chapter 8: Securing Your Search Results

    // Forms Authentication
    using System.Web;
    using System.Collections.Specialized;
    using System.Collections;
    // Data Access
    using System.Data;

Next, you need to implement the ISecurityTrimmer interface. You do this by adding the following code to your class definition:

    class CustomSecurityTrimmer : ISecurityTrimmer

The sample then declares a few variables that you will use later on. The first one is the number of items to check before throwing an error. This variable will be updated by the value passed as part of the trimmer registration. The second is an array that will hold the role names that you will parse from the role names passed into the registration. You will use these role names later in the SQL Server query that you write, to figure out whether users are members of the roles for the security trimming. Finally, you declare a string that will contain the comma delimited string from the registration.

    // Number of items to check for
    private int intNumItemsToCheck = 100;

    // Array of Role Names
    private string[] arrRoleNames;

    // Comma delimited names of roles passed in from reg
    // Used in SQL Server query
    private string strRoleNames;

Once this is done, you need to implement the methods for the ISecurityTrimmer interface that you discussed earlier: Initialize and CheckAccess. In the Initialize method, the code takes the passed in parameters as part of the registration and parses them into the variables in the code.

    public void Initialize(NameValueCollection trimmerProps,
        SearchContext searchCxt)
    {
        if (trimmerProps ["NumItemsToCheck"] != null)
        {
            intNumItemsToCheck = Convert.ToInt32 (trimmerProps ["NumItemsToCheck"]);
        }

        if (trimmerProps ["Roles"] != null)
        {
            // Parse the comma delimited string
            strRoleNames = trimmerProps ["Roles"].ToString();

            arrRoleNames = strRoleNames.Split (new char[]{',',''});
        }
    }
Chapter 8: Securing Your Search Results

For the CheckAccess method, you will need to step through the different sections of the code, since different things are happening as part of it. First, you need to check to make sure that there were some results passed before you start checking permissions on them as shown below:

```csharp
public BitArray CheckAccess(IList<String> crawlURLs,
IDictionary<String, Object> sessionProperties)
{
    if (crawlURLs == null || crawlURLs.Count <= 0)
    {
        throw new ArgumentException("There are no documents returned");
    }

    BitArray retArray = new BitArray(crawlURLs.Count);

    Next, the code will check to see whether it has crawled the maximum number of URLs for the session, so it does not crawl thousands of documents. Since your trimmer can be called multiple times, you need to persist the number of items that you have checked between calls to your code. The way you persist the value is by using the sessionProperties Dictionary object. The sessionProperties property allows you to persist properties for your security trimmer. You’ll compare the number you persist, and the number you need to check from the current call, to the method with the number that the registration said you should error out. The following code implements this functionality:

    if (!this.CheckCurrentCount(sessionProperties, crawlURLs.Count))
    {
        throw (new PluggableAccessCheckException("Access Check Limit Reached"));
    }

    private bool CheckCurrentCount(IDictionary<String, Object> sessionProperties, int numChecks)
    {
        Object currentCount;
        sessionProperties.TryGetValue("currentCheckCount", out currentCount);
        if (currentCount == null)
        {
            sessionProperties["currentCheckCount"] = numChecks;
            return (true);
        }

        int currentCountInt = Convert.ToInt32(currentCount);
        currentCountInt += numChecks;
        sessionProperties["currentCheckCount"] = currentCountInt;
        if (currentCountInt <= intNumItemsToCheck)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

Next, you need to implement the core functionality of your access check. This is where your code will differ, depending on how you want to implement your security checks and the type of security you are using for your search center and queries. The first thing you need to do is get the username of the person
performing the query. Again, the way the query is performed and accessed will determine how you do this. If you are using Windows Authentication, you will use a different method than if you used FBA. The code below shows you how to do both methods. The Windows Authentication code is commented out since this code is written to work with FBA. Remember that the authentication will not be the authentication you used to crawl the site, but instead the authentication you used to access the search center or search object model. Effectively, the crawl account you use is probably different from the user performing the query. For example, if you crawled an FBA site, but you are logged into a Windows Authentication site for searching, the username will be your Windows Authentication account. Also, in my testing I found that SharePoint sometimes puts a double slash between the domain and username. You will want to check and see whether your implementation performs in the same way, since you will have to account for this in any username parsing code that you write. This could be just the display in Visual Studio, but you will want to make sure that your code can handle this situation if it happens.

//For Windows authentication, uncomment the next line:
//string strUser = WindowsIdentity.GetCurrent().Name;
//For Forms authentication, uncomment the next line:

The next section of code implements all the database code to access the security database in SQL Server to query for the roles and users. Since this is standard SQL Server code, I will not list it here, but the general gist of it is to create a dynamic query string to figure out the roles that the person registering the trimmer passed in, and then to find all users in those roles. You could also use the membership API to do this check. The code sets a Boolean if the user is found in one of the roles, so that you can allow the search result to be returned.

If the user is in the role, then the code will set the index of the BitArray that corresponds to the search URL index. There are a couple of things you should know about how trimmers get the URLs. First, the URLs come in an IList. The URLs are formatted using the protocol handler notation, so HTTP URLs begin with http; however, SharePoint datasources start with sts3://, and BDC URLs start with bdc2://. This is important to know, since you may check for the datasource of the content when trimming your URLs. Second, you will need to understand how to open the different URLs using the object models for their particular datasource. For example, if you want to access a SharePoint site, you will need to parse the sts3:// URL. This URL will contain things like the siteID, webID, and listID. You will use common string manipulation to do this.

The following code literally flips the bit on the user, if they are a member of one of the specific roles, so they have access to the content:

while (sqlReader.Read())
{
    if (sqlReader[0].ToString() == strUser)
    {
        //User found, break while loop
        bUserInRole = true;
        break;
    }
}

for (int x = 0; x < crawlURLs.Count; x++)


Chapter 8: Securing Your Search Results

```c
{
    if (bUserInRole == true)
    {
        retArray[x] = true;
    }
    else
    {
        retArray[x] = false;
    }
    return retArray;
}
```

**Signing Your DLL and Adding It to the GAC**

In order to have SharePoint work with your trimmer, you have to sign your assembly. In addition, you will need to add it to the GAC. Remember that, if you have multiple query servers in your farm, you need to perform these steps on all your query servers.

To sign the DLL, you can use the properties page in your VS project and, under the Signing tab, use the Sign the assembly option. Create a new key using the `< New ... >` option. Then, build your assembly.

Once you have done these steps, you need to add your assembly to the GAC. Leverage the `gacutil` command-line utility to do this. Your command should look something like `gacutil /if <path to your assembly>`. This will overwrite any existing registrations that you have. If you used a WSP file for your deployment, SharePoint will do this for you.

**Registering your Customer Security Trimmer**

Now that you have the assembly in the GAC, you can register your security trimmer. `STSADM` has been extended with three new commands to support custom security trimmers. The first way to view listed trimmers is to use the `listregisteredsecuritytrimmers` operation of `STSADM` and pass it the name of the SSP you want to query. For example, the following will query for all security trimmers on the SharedServices SSP – `STSADM -o listregisteredsecuritytrimmers -ssp SharedServices`.

To create a new registration for your trimmer, there are a couple things you need to do before registering it. First, you need to create a crawl rule in SharePoint to which you want to attach your trimmer. For my FBA example earlier, you could create a crawl rule that logs on to the FBA site that has your secured content. Figure 8-1 shows creating a crawl rule for the trimmer.
Next, you need the public key token value from your signed assembly in the GAC. The easiest way to get this is to use Windows Explorer and navigate to drive:\windows\assembly. From there, find your assembly, right-click, and select Properties and then find the token on the General tab. Figure 8-2 shows what the property page looks like.
Chapter 8: Securing Your Search Results

The next step is to register your trimmer using STSADM. You need to call the registersecuritytrimmer command and pass in a few parameters. The first is the SSP you want to register. Second is a unique integer ID for the trimmer. You will want to remember this, since you will want to use this ID to remove the trimmer later or replace it. You can always retrieve the ID by listing all the trimmers on your SSP as well. The third is the typename. The fourth is the crawl rule for which you want to register the trimmer. Finally, you can optionally pass any configuration properties that you want to pass to your trimmer. In the example that follows, you pass in the number of items to check before erroring out and the names of the roles you want to have access to the content that the crawl rule specifies.

    stsadm -o registersecuritytrimmer -ssp SharedServices -id 1 -typeName "CustomSecurityTrimmerSample.CUSTOMSECURITYTRIMMER, CustomSecurityTrimmerSample, Version=1.0.0.0, Culture=neutral, PublicKeyToken=4a3b0a652cb43c64" -rulepath http://moss:7002/* -configprops NumItemsToCheck=300-Roles=Salesforce,Suppliers

You can also remove your trimmer by using the unregistersecuritytrimmer command and passing the SSP name and ID of the trimmer that you want to remove. For example, the command STSADM -o unregistersecuritytrimmer -ssp SharedServices -id 1 would remove the trimmer with an ID of 1 from the SharedServices SSP.

To make your trimmer work, you will need to perform an IISRESET on your machines after registering the trimmer, and also perform a full crawl using your crawl rules where the trimmer is registered. After that, you’re all set to try out your trimmer.

You can also register your trimmer programmatically using the administration object model for search. Given that it is probably easier for you to script STSADM than to write custom code, we won’t cover that technique here, but the code to perform the programmatic installation is very straightforward.

Debugging Your Customer Security Trimmer

Of course you would love to write perfect code, but sometimes we make mistakes. With your custom security trimmer, you can debug your code by using Visual Studio. The process is similar to the way you would debug a web part or other server-side SharePoint code. Please note that you cannot perform remote debugging, so you will have to run the debugger on your SharePoint Server. This may not work well in a production environment, so you may want to add logging to your trimmer code, so that you can debug your log, rather than having to attach a debugger to your production servers.

To debug your trimmer, set your breakpoints in your code. You will want to attach to the w3wp.exe process and then place Visual Studio in debug mode by pressing F5. Once you have done that, fire up your browser or search application, and perform a search that will trigger your security trimmer to run. Your breakpoints should be hit, and you should be able to debug your code.
BDC Security Trimming

The Business Data Catalog is an important piece of technology in SharePoint, since it provides access to other systems for the crawler in an easier way than writing your own protocol handlers. With this simplicity, you do trade off some of the customization that you get with writing your own code; however, when it comes to security, the BDC does a fairly good job at providing you with a rich set of flexible options from both an authorization and an authentication standpoint. Before reading this section, you should familiarize yourself with the BDC content in Chapter 5.

Authentication and the BDC

When it comes to the BDC security model, the BDC offers two authentication modes: Trusted Subsystem and Impersonation/Delegation.

In the trusted subsystem mode, the web server in which the BDC is running logs in to the back-end system, using a fixed identity. The benefit of this model is that you can use database connection pooling. You may have only one account given to you by the back-end-system administrators, and it is a less complex model than the impersonation model.

With impersonation, you have the BDC impersonate the user. There are two ways that this can occur. The first is by using pass-through authentication. This means that the server in which the BDC is running passes the credentials of the current user to the back-end system. The main issue with this method is the dreaded double-hop problem, where credentials can hop once from the client to the server running the BDC, but can’t hop again from the BDC server to the back-end system. The way to resolve the double-hop problem is to enable Kerberos delegation in Windows Server. With pass-through authentication you cannot use a two-factor authentication such as smart cards or other types of authentication that require an extra authentication mechanism beyond username and password. RevertToSelf does not suffer from this issue, since it is single-hop. You could implement SSO to get around the double-hop problem, as well.

The second type of impersonation that you can do is called RevertToSelf. Rather than just passing the credentials, the server in which the BDC is running will stop impersonating the user, so that the application pool identity is used, and then attempt to perform its operations. The permissions of the application pool identity are used to perform the operations, so any access restrictions on that identity will be enforced.

You can use the SharePoint SSO store with the BDC to allow the BDC to pull credentials from the store. In addition, you can even provide a second set of credentials to the BDC, if you need application-level authentication for your LOB system. Refer to the SharePoint SDK for more information on implementing SSO in SharePoint.
Chapter 8: Securing Your Search Results

**BDC and Search**

The BDC allows you to crawl, index, and query entities from the back-end system. By default, the BDC provides only entity-level security. This means that all users that have access to the entity store have access to query on all the entity instances in that store. Let’s take a real-world example to illustrate this. Imagine that you have a CRM system and you have all your leads and customers in that system. If you give your users access to the system through the BDC, any user can search and see any entity in that system, regardless of whether they have application-level permissions to be able to see those instances.

This is where the custom security trimming talked about earlier comes into play. The BDC implements the `CheckAccess` method that you learned about earlier to allow you to trim search results that come back from a line-of-business (LOB) system. The LOB system will have to provide a method that tells the query processor which users have permissions to which entities. You can use the `MethodInstance` type called `AccessChecker` to tell SharePoint to which LOB method to pass the entity instances, so that the LOB system can decide whether the user has permissions.

Defining your security trimming in your BDC definition is a no-code solution, since you do not have to fire up Visual Studio. You have to figure out how to get permissions from the back end, but that approach normally is to call a stored procedure, web service, or some other method to retrieve the permissions. The BDC built-in security trimmer is not your only option to trim results. You can use a code-based security trimmer solution like those discussed earlier in the chapter. The main difference between the two will be that when you register the solution, you will point to your code-based solution, rather than to the built-in BDC security trimmer implementation.

One thing to note is that security trimming only works with search results. If you want to trim the BDC web parts, you will want to use the user context filters and have your queries to the back-end system trim based on BDC filters. You will want to make sure that you do not make the filters user controllable. Otherwise, one user could gain the ability to see the data for another user by customizing the filter.

**Performance Implications**

There are a couple of performance implications that you should be aware of when you want to use the BDC security trimmer. First, you should make sure that your back-end system is ready for the extra load for all the access checks that the BDC will send to your back-end system. Depending on how well you write your code, you can either scale well or make your back end very slow, since a lot of results could come back that need to be individually checked by SharePoint. Second is that SharePoint will attempt to perform multiple checks in a single call by passing multiple IDs. If your security checker does not support multiple IDs, then SharePoint will spawn multiple threads to check the access. This can cause a lot of strain on your back-end system, so you need to be prepared for that extra load.

**Writing Your Custom Security Trimmer**

To get started with writing your BDC custom security trimmer, you will need to modify your BDC application definition to include some new methods and parameters. If you have already installed your application definition file, you will want to update the version number of that file, so that SharePoint will know that it is a new version of the file.
To see how to write a trimmer, you will be using the Pubs sample database. As part of the sample, you will filter out particular authors based on user permissions stored in the table. Figure 8-3 shows the authors’ table in SQL Server Management Studio. Notice that the last column contains the name of the user who is allowed to access the record.

The way you implement your trimmer is to implement a method instance in the BDC of type AccessChecker. The BDC security trimmer executes the Entity.CheckAccess method when the search engine returns results. In turn, this method calls your AccessChecker functionality, defined in the BDC application definition, to check the access for the entities returned by the search. Your method needs to pass back whether the results should be displayed to the user.
The modifications you need to make to the application definition file is to add a new method to your entity. This method encapsulates your entire security trimming functionality. In the sample code, this method is called UserRightsForAuthors. As part of the method, you need to set some properties. The properties will contain the command text to send to SQL Server in order to retrieve the user rights for the current user executing the query against the BDC. You will see how you get the current user context in a couple of minutes.

The command that you will send will query the database to retrieve the current author, based on the ID passed into the method and whether the current user is listed in the UserName column (which lists the user who should have access to the entity). You will get a count of the number of rows returned, which will be 0 if the user does not have permissions and a 1 if the user does. The beginning of the XML code for the method definition is:

```xml
<Method Name="UserRightsForAuthors">
  <Properties>
    <Property Name="RdbCommandText" Type="System.String">
      SELECT CAST(count(*) as bigint) as Rights
      FROM authors
      WHERE
        au_id = @au_id and UserName = @currentuser;
    </Property>
    <Property Name="RdbCommandType" Type="System.Data.CommandType, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">Text</Property>
  </Properties>
</Method>
```

You need to set up a user context filter. This filter will pass the username of the person executing the query, so that you can use it in your SQL query. This filter will feed the parameters into your method. Try to remember your naming convention across all these definitions; put the names in your appdef, since it can get confusing when you have to define multiple filters, parameters, and other settings with similar names. One of the easiest mistakes to make is to misname something in your definition; then, your trimmer will not work. The definition for the user context filter is:

```xml
<FilterDescriptors>
  <FilterDescriptor Type="UserContext" Name="currentuser" />
</FilterDescriptors>
```

After the filter, you need to define the parameters that your method will both take in and return. As you can see from the earlier SQL query, you need to get the ID of the author and the current user’s name as IN parameters, and you need to return the integer that indicates the user’s rights to the item. The author ID is a straightforward typedescriptor for the BDC, since you are just passing in a string with the identifier information.
Chapter 8: Securing Your Search Results

The current user typedescriptor is complex because you have to tell SharePoint the name of the associated filter from which to retrieve the username. You will see in the code that the filter you use is the filter just discussed; it returns the user context for the query.

For the return parameter, you will see that the type is a SQL datareader. Because you are returning results from a database, you have to use this type. Also, even though you are returning a single row, you still have to set the IsCollection attribute to true because you will return a record stream, even for a single record. Since it is a collection, you need to define children type descriptors. These children are the actual records. You set the IsCollection to false by not defining it, which is the default value. The code for this is:

```xml
<Parameters>
  <Parameter Direction="In" Name="@au_id">
    <TypeDescriptor TypeName="System.String"
      IdentifierName="au_id" Name="au_id" />
  </Parameter>
  <Parameter Direction="In" Name="@currentuser">
    <TypeDescriptor TypeName="System.String"
      AssociatedFilter="currentuser" Name="@currentuser" />
  </Parameter>
  <Parameter Direction="Return" Name="Rights">
    <TypeDescriptor TypeName="System.Data.SqlClient.SqlDataReader, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
      IsCollection="true" Name="AccessCheckRecord">
      <TypeDescriptors>
        <TypeDescriptor TypeName="System.Data.IDataRecord, System.Data, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" Name="AccessCheckRecord">
          <TypeDescriptors>
            <TypeDescriptor TypeName="System.Int64" Name="Rights" />
          </TypeDescriptors>
        </TypeDescriptor>
      </TypeDescriptors>
    </TypeDescriptor>
  </Parameter>
</Parameters>
```

The final piece that you have to define is the MethodInstance. The MethodInstance describes how to use the method. In your case, the type of the MethodInstance will be AccessChecker. You could have other types such as a Finder, GenericInvoker, SpecificFinder, or other types of MethodInstance objects. You must also define the name of the parameter that the method will return, the type of that parameter (by specifying the type name), and the level in your type definition hierarchy (where the parameter type definition sits).
Chapter 8: Securing Your Search Results

<MethodInstances>
    <MethodInstance Type="AccessChecker">
        ReturnParameterName="Rights" ReturnTypeName="Rights"
        ReturnTypeDescriptorName="Rights"
        ReturnTypeDescriptorLevel="2" Name="UserRightsForAuthors" />
    </MethodInstances>

That’s all you need to do to get your security trimmer application definition set up. The harder parts are deploying and debugging your trimmer. Let’s spend some time looking at both these steps.

**Deploying Your BDC Security Trimmer**

To deploy your trimmer, you will first need to create the application in the BDC that contains your modified code and the `AccessChecker` method. You could use the SSP administration in Central Administration, or you could deploy your application definition via code. Once you have your application loaded, you should create your crawl rule. The path for the crawl rule should start with `bdc2://`, which is the reserved path for BDC results. If you use `bdc2://*`, then all BDC results will pass through the trimmer. If you want to limit the results to just your BDC application, you can figure out the system instance ID and entity ID of your application by looking through your crawl log, or by looking at the URL when you access your instance in the administration interface. Your URL will then look something like `bdc2://*<systeminstanceID>/<entityID>` where the `systeminstanceID` will be an integer and the `entityID` will be an integer. You can use any combination of wildcards for your rule path with these integers, depending on the needs of your application.

Once you have your crawl rule set up, you need to register the security trimmer using `STSADM`. You will use the same `registersecuritytrimmer` command that you used earlier. Rather than specifying the assembly you created and placed in the GAC, you will register the built-in SharePoint BDC trimmer, which is named `Microsoft.Office.Server.ApplicationRegistry.Search.QueryProcessorSecurityTrimmer`, `Microsoft.SharePoint.Portal`. Besides that one change, the rest of the command is the same as you saw earlier, and you need to pass the SSP, an ID for your trimmer, and the rulepath. A complete example of a register command is:

```
"c:\program files\common files\microsoft shared\web server extensions\12\bin\
stsadm -o registersecuritytrimmer -ssp SharedServices1 -id 99 -rulepath bdc2://*
```

Once you have registered your security trimmer, you can create your content source to define the crawl of your BDC application. There is a known issue that if you create your content source before you register the trimmer, your trimmer will not be called. So, if you make a mistake and need to recreate your registrations and application definitions, make sure to undo, and then redo, the steps outlined here to make sure that your trimmer gets called.
Chapter 8: Securing Your Search Results

After your content source is created, perform a full crawl on the new content source. This will have the BDC crawl the content and also will acknowledge that you have custom security trimming enabled on that content.

Next, go to your search center and perform a search. You should see all results from the BDC to which you have rights. Sign in as a different user and perform the same search, and different results should appear. Let me show you an example of what this looks like.

You’ll perform a search using one account that has permissions for all the authors except one. Figure 8-4 shows the search for a particular author where the user has permission to see the record.

Figure 8-4: Searching for an author that you have permission to see
Chapter 8: Securing Your Search Results

Next, perform a different search looking for an author for whom you do not have permission. Notice how you do not get any BDC results in the user interface (see Figure 8-5).

![Figure 8-5: Searching for an author that you do not have permissions to see](image)

Now, log on as a different user, who does have permissions to see the author that the original searcher did not. In the screenshot shown in Figure 8-6, you will see that author appear for this user. Security trimming is working!
Debugging Your BDC Security Trimmer

As always, developing is one piece of your application, debugging is the other. Unfortunately the debugging experience for using the built-in BDC security trimmer is not as rich as developing your own, since you can’t kick the trimmer into debug mode in Visual Studio and step through your code. Instead, you will have to look through your SharePoint Unified Logging Service (ULS) logs to see if you are getting issues. To make reading the logs much easier, I recommend that you download the Log Viewer feature from Codeplex at the following location: http://www.codeplex.com/features/Release/ProjectReleases.aspx?ReleaseId=2502. Figure 8-7 shows using the Log Viewer feature.
Chapter 8: Securing Your Search Results

Figure 8-7: The Log Viewer for reading the SharePoint ULS logs

The categories in the log you want to look at are the Search Server 2008 and Business Data categories. These different categories will tell you what happened with your Search Server 2008 crawls and your BDC interactions. The following shortened example from the log shows the BDC crawling your authors’ datasource by executing the SQL query provided and retrieving the results:

```
12/21/2007 13:47:44.87 mssdmn.exe (0x14F4)0x1458  SharePoint Portal Server
Business Data                   71tw  High      Initialized DbSystemUtility for
LobSystemInstance with Id '1389'
12/21/2007 13:47:44.87 mssdmn.exe (0x14F4)    0x1458
SharePoint Portal Server        Business Data                   71tx  High      Db
AuthN Mode: PassThrough
12/21/2007 13:47:44.87 mssdmn.exe (0x14F4)    0x1458
SharePoint Portal Server        Business Data                   71ty  High      Db
Provider: SqlServer
12/21/2007 13:47:44.87 mssdmn.exe (0x14F4)    0x1458
SharePoint Portal Server        Business Data                   71q0  High
Checking for current connection to SystemInstance: pubs_Instance
12/21/2007 13:47:44.87 mssdmn.exe (0x14F4)    0x1458
SharePoint Portal Server        Business Data                   71op  High
Delegating current windows identity
12/21/2007 13:47:44.87 mssdmn.exe (0x14F4)    0x1458
```
Chapter 8: Securing Your Search Results

One of the things that you should look for in the log is errors. The BDC will silently fail by putting the error information into the log, and you may find yourself scratching your head wondering why your search is not working. To help you with the most common errors, we put together a list.

First, make sure that your SQL query actually works for all scenarios. Test the query in management studio and plug in all the different variations of the expected parameters to make sure that your code can handle it (things like all caps, double slashes, or different input types). The most common scenarios for your BDC definition and crawl failing are problems with your queries, whether it’s SQL or a web service.

Second, make sure that your datatypes are the right type for your mapping. Another common error is to attempt to return a 32-bit integer when a 64-bit one was expected. Make sure that your query is returning the expected data types for your application definition.
Chapter 8: Securing Your Search Results

Third, check the logs to see whether the parameter values you are expecting are, in fact, the parameter values you are getting. For example, when the BDC passes the username for a person using Windows authentication, the domain name is always capitalized, as in $\text{DOMAIN}\backslash\text{username}$. If your code was not expecting the capitalized letters and does the wrong thing because of it, you may get incorrect results.

Fourth, make sure to give the right user rights to your users to search and use the BDC. These are the Selectable in Clients and Execute permissions. The minimum permission needed is the Selectable in Clients permission, but you may want to add the Execute permission, so that users can execute methods on the instances from the BDC.

Securing Your Search Server 2008

Beyond securing trimming, you will want to make sure that you are securing your Search Server 2008 in the correct way, so that you are not exposing content to users who may not be allowed to see the content. Selecting the right accounts in which to run your services and to which to give access to your content will make or break your deployment.

Default Content Access Account

You must make sure that the Default Content Access account is a low privileged account on your farm. If you make this account a highly privileged user, your user may get results back that are unintended. For example, with SharePoint versioning, every version uses the same URL, so that a highly privileged user will crawl both published and unpublished versions of the document. Therefore, when users search for content, they can get the unpublished and possibly unapproved content back in the search results. Make sure not to make the account a member of the farm administrator group.

You can modify the Content Access account using crawl rules so that you can use a different Windows account, forms-based authentication, or other types of authentication. This will not use the default Content Access account, but the accounts that you specify, instead.

Single Server Deployment

A single server deployment is your easiest deployment. With this deployment, you will want to use the Network Service and Local Service accounts. For example, the Server Farm account, which includes the app pool for central admin and the account in which the timer service runs, the SSP Web Application Pool and the SSP Service account, should use the Network Service account. The Microsoft Office SharePoint Server Search account and the Default Content Access account should use the Local Service account, unless you are crawling remote content. If you are crawling remote content, then you will want to use a Domain account.

Server Farm Deployment

With a farm deployment, you will want to make some changes to the accounts that you use. First, you will want to make sure that your Server Farm account has the right privileges in SQL Server and is an owner of all the databases in your farm. Second, you want to have your SSP web application pool and SSP Service accounts own the SSP databases in SQL Server, read/write on all your content databases,
and have read access for the configuration and central admin content databases. For your Microsoft Office SharePoint Server Search account, you want read/write on all content databases and the Microsoft Office SharePoint Server search database, and read-only access to the configuration database.

**Conclusion**

In this chapter, you have seen the security architecture for SharePoint Search, how you can trim results (for all search results by writing a security trimmer, or for BDC results using the built-in security trimmer), and the settings for securing your Search Server 2008. Security is a critical piece of any search infrastructure, so you will want to make sure that you are testing and retesting the different security settings and search results for different accounts and permissions, before deploying your search solutions to your end users.
Customizing the Search Experience

Search is all about presenting relevant results in a compelling user interface. To do this, you need a rich index and query engine and a complementary rich and extensible user experience to present results. SharePoint provides an out-of-the-box search result experience through the use of web parts that work together and are customizable. By breaking the user interface into a number of different web parts, you can quickly change the user’s search experience whenever you need to. Plus, you can add your own new web parts to the search experience based on your business needs. This chapter will step you through the search experience in SharePoint and how to customize that experience.

Describing the Flow of a Typical Search

The user interaction with search in SharePoint typically starts with a user bringing up the Search Center either directly or by typing a query into a browser search box, performing that query, and seeing the results. There are a lot of complex things that happen as that query is processed: Results are returned, web parts are instantiated, data is shared across them, transformations happen, and results are displayed. Let’s step through the process in a little more detail, so you can understand where the customization points can happen.

A search query can come from a number of different sources. Depending on the source, the query could be sent using HTTP GET, HTTP POST, or sent by the object model. The query is executed, and SharePoint creates a hidden object to cache and share the results for use by all the different web parts on the page. The web parts perform Extensible Style Language Transformation (XSLT) transformations on the data that is given to them in an XML format. Then, they display their results in the user interface. If you are not familiar with XSLT, you will be, after customizing SharePoint search, since it is used extensively. XSLT, for those of you who have not used it before, is the means to describe a transformation for your XML data. Most commonly, that transformation results in outputting HTML for display of the XML, but XSLT could also translate XML from one XML format to another. Figure 9-1 shows a standard results page in SharePoint.
Chapter 9: Customizing the Search Experience

There are 11 out-of-the-box web parts in your Search Center. Table 9-1 describes the different web parts and what they do:

Table 9-1

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search Box</td>
<td>The search box in which you can type your search query.</td>
</tr>
<tr>
<td>High Confidence</td>
<td>The high-confidence matching results are displayed in this web part. Out of the box, only people results are displayed here, since the mapping is to people centric properties. You can change your properties to map to the HighConfidenceMatching property so that any custom property can be mapped for a high-confidence result.</td>
</tr>
<tr>
<td>Core Results</td>
<td>The main web part for your results page. It shows results ranked by relevance by default.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Returns the statistics for the query, such as number of results displayed, total results, and elapsed time.</td>
</tr>
<tr>
<td>Pagination</td>
<td>Controls the paging of search results.</td>
</tr>
<tr>
<td>Action Links</td>
<td>The actions for the results are displayed here. You can change how the search results are ranked, create alerts, or subscribe to a search RSS feed. You can use XSLT to add new actions to the display.</td>
</tr>
</tbody>
</table>

Figure 9-1: The Search results page

(Continued)
### Table 9-1 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching Keywords</td>
<td>This web part shows the matching best bet results that you have identified</td>
</tr>
<tr>
<td>and Best Bets</td>
<td>that match the search and any keyword or high-confidence result matches.</td>
</tr>
<tr>
<td>Search Summary (Did you mean?)</td>
<td>Displays a summary of the query such as keyword and scope used. If any</td>
</tr>
<tr>
<td></td>
<td>corrections should be made to the query such as misspellings, suggestions</td>
</tr>
<tr>
<td></td>
<td>are displayed in this web part.</td>
</tr>
<tr>
<td>Advanced Search</td>
<td>Allows you to perform advanced searches by customizing the targeted content</td>
</tr>
<tr>
<td></td>
<td>and properties for your query.</td>
</tr>
<tr>
<td>Federated Results</td>
<td>Displays your federated results for a federated datasource.</td>
</tr>
<tr>
<td>Top Federated Results</td>
<td>Allows you to display the top results from multiple federated datasources.</td>
</tr>
</tbody>
</table>

Figure 9-2 shows the same results page, but with the different web parts identified on the page.

![Search web parts identified](image)

**Figure 9-2: The Search web parts identified**
Chapter 9: Customizing the Search Experience

SharePoint also supports the use of tabs so that you can break out your search experience. It is up to you how you want to leverage your tabs, and whether the tabs represent different content sources, different user interfaces, or even different queries. The typical tabbed user interface has different tabs for different content sources. For example, you may want to split your unstructured, people and line-of-business (LOB) results into distinct tabs, so that users know where the data is coming from, rather than seeing the results all in a single user interface.

By combining the web parts, tabbed interface, data, and transformations, you create your search experience. This experience can be customized in a number of different ways, ranging from no-code solutions, to completely rewriting the search experience using the object model. The rest of this chapter will step you through the different customization options.

**Customizing the Search Center — No Code**

First and foremost, the Search Center is a site template, and it builds on the SharePoint web part framework. This means that you can use a lot of inherent SharePoint features to customize your search experience without having to write a line of code. For example, if you want multilingual search, you can use variations with the Search Center. Since it is just a site, you can have multiple Search Centers in your site collections.

Another simple way to customize your search results is to customize the web part properties for the search web parts. The Search Center is just a number of web parts put together on a page. For example, you can remove or add web parts to your search pages. Also, you can modify the properties for the web parts to make the search experience different for your users. Let’s step through some properties you can modify on the different web parts.

The web part that has many properties that you can change to make the search experience different is the Search Box web part. From this web part, you can modify whether you want to show scopes, and with a number of different options. Scopes allow you to refine your queries, with options such as All Sites, People, or custom scopes. The Search Box web part also allows you to customize the query that is sent to the server, by allowing you to specify additional terms to append to all queries. This is useful if you want to perform contextual searches, where you know that your search page is sitting on a site that should be refined using particular terms. Finally, the Search web part allows you to skin its user interface (UI) and result settings just using properties, by providing the ability to change out the images that appear, such as the Search button, the Advanced Search link, or both the Advanced Search and page results’ URLs. Figure 9-3 shows the property pane for the Search Box web part.
Chapter 9: Customizing the Search Experience

To change the way your Search Center works, you may want to customize the properties for the Core Results web part. This web part is where all the results from your search appear. With the properties in this web part, you can change the number of results per page, the total number of results, the number of sentences in the summary, whether or not to collapse duplicates, and a number of other options. One important option to point out, that this web part supports, is whether to turn stemming on or off. Stemming finds all the derivatives of a word, such as run, running, and ran, and can even do similar words such as electric and electricity. The final property of the Core Search Results web part that you can modify, and we will use this property extensively later in the chapter, is XSLT, used to format the Results user interface for your search results.

**Tab User Interface**

Beyond modifying web part properties, you can also modify the tabbed user interface in order to customize the query or results user interface for Search. The most common use of tabs is to display different content sources using different tabs. For example, you may want to display your unstructured datasources in one tab, your people results in another tab, and your LOB results in a third tab. This is not the only way that you can use tabs, since they are completely customizable.
Chapter 9: Customizing the Search Experience

Tabs are implemented as web pages contained in a SharePoint list. There are two lists — one for your search query page and one for your results page. You can find these lists under your Search Center site, or, if you click the Edit Tab button in the user interface, the list will appear, as shown in Figure 9-4.

![Figure 9-4: The Tab-Editing user interface](image)

You can quickly add a new tab by using the New Tab option in the UI. You will have to either point the new tab properties to a new search page or create a new page. You can create the new tab without having the new page created. When you create a new tab, you will be prompted for the name of the tab, the page to be displayed, and an optional tooltip for the tab, as shown in Figure 9-5.
Once you have typed in these properties, you have to create the page that will serve back the results for your tab when someone performs a search in it. The way to do this is to create a new page using the Site Actions menu in your Search Center site. The create page action is site sensitive, so it knows the types of pages you can create in a Search Center site. You should first create your search page for your new tab, and then your search results page. Using this user interface, you only need to supply the title of your page, the URL you want, and a description. The Create Page user interface is shown in Figure 9-6.

After you create your search page, you should edit the page, and, in the properties for your search box, under the Miscellaneous section, you should modify the Target search results page URL property, so that it points to the search results page you created using the Create Page functionality we described earlier.

In your results page, you will want to modify the same property, so if someone types in a search in the results page, the search page calls itself, rather than the default search page. If you want to customize to just a particular scope, as we discussed as one of the reasons to use scopes earlier, just modify the Scopes property in your Core Results web part, and type in the name of the scope to which you want to limit the results. For example, you could limit it to customers, partners, products, file shares, or whatever you have defined as one of your scopes in the search administration.

Also, you will want to make sure that your new tab appears on all your pages, so double-check that when you click in all the search pages and search results pages for all your tabs, that your new tab appears in all these pages.
Chapter 9: Customizing the Search Experience

**Thesaurus and Synonyms**

With SharePoint, you can expand the vocabulary that the search engine understands by using the thesaurus feature for Search. With the thesaurus, you can replace words with other words when a user searches, and you can also provide expansion sets that include stemming. For example, if you wanted to replace an acronym with the full word, such as IE with Internet Explorer, you would use the thesaurus file. You can also expand the search based on similar words, so if you wanted to have the search expand whenever one of the words in an expansion set is hit, such as IE, Internet Explorer, or web browser, you also use the thesaurus in SharePoint.

By default, SharePoint ships blank thesaurus files for every language that the server supports, and then some, to make a total of over 40 languages that are supported, plus a language-neutral thesaurus file that applies to every language. You will find the thesaurus, and noise word files for that matter, under the program files\Microsoft Office Servers\12.0\Data\Office Server\Applications\ (GUID for SSP)\Config folder. Often, folks ask how to quickly get the Globally Unique Identifier (GUID) for the SSP that they’re working with. Some folks recommend going into central admin to edit the SSP to see the SSP ID on the querystring. The better way to get the GUID is to edit the properties for the SSP and find the name of the database for the SSP. On the server, go into the registry, and under HKEY_LOCAL_MACHINE\Software\Microsoft\Office Server\12.0\Search\Applications, expand the GUIDs and look under each GUID at the ResourceManager subkey. Compare the value of
the search subkey with the database server and name that you saw in central admin. If they are the same, you have found your SSP GUID in the parent key!

Once you find the folder, you will see a lot of files that start with tsxxx.xml. These are the thesaurus files where the xxx is the name of the language, such as enu for English, deu for German, and neu for the neutral file. You will want to modify the language specific files with any specific language nuances that you want to implement, but the majority of your changes will probably be in the language neutral file, especially for industry acronym expansion or other types of non-language-specific changes you want to make.

When you open any of the XML files contained in the directory, you will find that they are commented out. Remove the comments and you can modify or start new in the file to make your changes. Just make sure not to remove the root XML node that contains the ID attribute, and the thesaurus node that points to the XML namespace schema file, which is tsschema.xml. Do not modify tsschema.xml in your folder. The sample thesaurus file that ships with SharePoint follows. You can turn on diacritic sensitivity by setting the number to 1 for that node.

```xml
<XML ID="Microsoft Search Thesaurus"
     <!-- Commented out
     <thesaurus xmlns="x-schema:tsSchema.xml"
     <diacritics_sensitive>0</diacritics_sensitive>
     <expansion>
       <sub>Internet Explorer</sub>
       <sub>IE</sub>
       <sub>IE5</sub>
     </expansion>
     <replacement>
       <pat>NT5</pat>
       <pat>W2K</pat>
       <sub>Windows 2000</sub>
     </replacement>
     <expansion>
       <sub>run</sub>
       <sub>jog</sub>
     </expansion>
     <!-->
     </thesaurus>
     </XML>
```

A couple of things to note. First, you can see both expansion sets and replacement sets in the sample. SharePoint supports both. With the expansion set, SharePoint will search for all the words in the set as synonyms. In the sample, results for IE, Internet Explorer, and IE5 will all be returned, as well as all results for run and jog. Just note that you shouldn’t put the same term in multiple sets; instead, combine sets into one big expansion if you need to.

You can also perform and weight stemming on your expansion set. The example snippet below weighs jog more than trot in the expansion set by setting the weight attribute on the nodes. SharePoint will lower or raise the ranking based on your weighting. In addition, there are double asterisks after jog, which indicates that SharePoint should perform stemming on the word. This means that jog, jogging, and jogged will also be included in the search query and results.
Chapter 9: Customizing the Search Experience

With replacement sets, SharePoint replaces the search term with the terms that you provide. This is useful if you want to replace acronyms with the full spelling of words, or find in your search reports that zero results are being returned for a term because of consistent misspelling. Replacement sets require you to put in pattern nodes using the pat tag, and then a substitution node, which is the substitution for the patterns. You can have a blank substitution if you want to return no results for the pattern.

Remember that in 2007, the thesaurus is case insensitive by default. In previous versions, you had to turn off case sensitivity by adding a node to your XML file that set the caseflag attribute of the case node to false. Here is an example:

```xml
<case caseflag="false"/>
```

With 2007, you do not need to do this, and if you add this flag, your thesaurus will not be loaded.

**Restarting the SharePoint Search Service**

Once you modify and save your thesaurus file, you will want to restart your SharePoint Search service for your changes to take effect. The easiest way to do this is to perform a `net stop osearch` and then a `net start osearch`. Once the service is restarted, your changes to the thesaurus file should be reflected in your SharePoint searches.

**Stemming**

A quick section on stemming is necessary, since lots of folks are confused about what stemming actually does in SharePoint. Stemming is the process of breaking down a word into its base form, (e.g., run is the base form of running), and then generating all conjugations of the base form (such as ran, running, run, or runs). Stemming is different from wildcard searches because wildcard searching is pattern matching. For example, if you put in run*, then you would get runt, runescape, run DMC, and anything else beginning with run. If you turn on stemming, you will get results that match all the conjugated forms run, running, ran, and runs. An exact match will be ranked higher than matching a conjugated form of the search term. In addition, the stemming is lexicon based, which means that you will also get irregular forms of a search term, such as mouse and mice or leaf and leaves.

Stemming adds some great abilities and relevancy to your searches, but there are some “gotchas” that you should be aware of if you use it. Turning on stemming will increase your index size and will also make your queries take longer, since the search engine has to generate the conjugations and perform searches against those conjugations. SharePoint does not support stemming in all languages, so turning on stemming in a language that does not support stemming will have no effect. Stemming is off by default, but you can turn it on in the properties for the Core Results web part.

Please note that SharePoint does not support wildcard searches but does support stemming. You will see double asterisks used sometimes, to show stemming, but not a single asterisk in queries, which would indicate a wildcard search.
Customizing People Results

Since people results are a little bit different from standard data results, because there are different web parts that render your people results, you will have to modify the properties for these web parts separately from the modifications that you make to your core result and other unstructured search web parts. There are two distinct web parts for people search: the People Search Box web part and the People Search Core Results web part. As the names imply, the Search Box Web part provides the search box to perform people search, and the Results web part returns the results. The Results web part derives from the Core Search Results web part and provides a different view by default from the Core Results web part. Rather than displaying the results via relevance, the People Results web part displays the results ranked by social distance, so that users can quickly assess who are the closest people in their social network related to the search term that they typed.

Since these are web parts, SharePoint provides a number of properties that you can modify to change the way the web parts work. Most of the properties are the same as those in the non-people search equivalents. The property for people search that you will modify, that is different from the properties in non-people search, is sorting, found in the people search core results; you can modify the sorting to be relevance rather than social distance. Social distance is the default sorting.

One other thing about people search of which you should be aware is that, when sorting by social distance, SharePoint breaks the results into groups called colleagues, your colleagues’ colleagues, and then everyone else. Colleagues are tracked through the My Site technology in SharePoint where you can add or remove colleagues, and you can also group colleagues together. You can add a new colleague manually or through the search results in SharePoint; however, there is some confusion on how colleagues are suggested to you as part of the My Site user interface. SharePoint works with the Office client, in particular Outlook, to detect new colleagues for you, based on whom you email, and who emails you. This only works with Outlook 2007, and if you go into Outlook and look at the add-ons inside of the Outlook client, you will see the colleague add-on. SharePoint also works with instant messaging (IM) applications, such as Windows Messenger or Office Communicator, to pull colleagues from your contact lists from IM.

SharePoint Designer Support

While this is not search specific, SPD (SharePoint Designer) makes a great tool to customize your search results, both in terms of modifying pages directly and when you work with the XSLT, as described later in the chapter. You can use SPD or Visual Studio to help you craft your XSLT rather than doing it by hand.

SPD will also add development speed to your customizations, since it allows you to set properties and make modifications much more quickly than just using the out-of-the-box web part property customizations.

Customizing your Search Results — XSLT

Sometimes when you are customizing search, the simple customizations that you can do with the web part properties are not enough to meet your business needs. This is when you need to turn either to XSLT or to code. With XSLT, you can perform a lot of powerful customizations by changing the UI, changing the properties that are rendered, rendering new interfaces, performing calculations, or creating
other types of customizations. Of course, you are limited in what you can do with XSLT, and if you run into a wall, the next step is to go to code. We will cover the coding aspects of the SharePoint Search APIs later in this chapter, but often, for the majority of your medium-complexity customizations, you can use just XSLT.

SharePoint ships an out-of-the-box XSLT template that it uses to display the search results. Since the results are returned to the results web part as XML, the XSLT is applied before displaying the final results to the end user. Before diving into how to alter the XSLT to learn how to modify, let us first take a look at the XML that comes back as search results and that will help make the XSLT easier to understand. Following is some sample XML that is returned by SharePoint for a search.

```xml
<All_Results>
  <Result>
    <id>1</id>
    <workid>1458</workid>
    <rank>880</rank>
    <title>Leading Bike Distributors.html</title>
    <author>Jesse Merriam</author>
    <size>270</size>
    <url>http://moss.litwareinc.com/SiteDirectory/mktg/Shared Documents/Leading Bike Distributors.html</url>
    <urlEncoded>http%3A%2F%2Flitwareinc%2Escom%2FSiteDirectory%2Fmktg%2FShared %20Documents%2FLeading%20Bike%20Distributors%2Ehtml</urlEncoded>
    <description></description>
    <write>1/8/2007</write>
    <sitename>http://moss.litwareinc.com/sitedirectory/mktg</sitename>
    <collapsingstatus>0</collapsingstatus>
    <hithighlightedsummary>Leading Bike Distributors  Litware Inc. is proud to announce that Leading Bike Distributors has decided to exclusively feature Litware's Draco product on all their new model bicycles. Truly a union of the best product in the world with the best manufacturing process.</hithighlightedsummary>
    <hithighlightedproperties>
      <HHTitle>Leading Bike Distributors.html</HHTitle>
      <HHUrl>http://moss.litwareinc.com/SiteDirectory/mktg/Shared Documents/Leading Bike Distributors.html</HHUrl>
    </hithighlightedproperties>
    <contentclass>STS_ListItem_DocumentLibrary</contentclass>
    <isdocument>1</isdocument>
    <picturethumbnailurl>_layouts/images/html16.gif</picturethumbnailurl>
    <imageurl description="Result of type: document">
      <_layouts/images/html16.gif</imageurl>
  </Result>
  <Result>
    <id>2</id>
    <workid>1213</workid>
    <rank>874</rank>
    <title>Bike Universe</title>
    <author></author>
    <size>0</size>
    <urlEncoded>http%3A%2F%2Flitwareinc%3A32901%2Fssp%2Fadmin%2FContent%2FReseller%2Easpx %3FResellerKey%3D347</urlEncoded>
  </Result>
</All_Results>
```
Chapter 9: Customizing the Search Experience

You will find when you look at the object model, that the XML returned through the object model is different from the sample XML you saw above. You may be wondering how you retrieve the XML for your own search results. The easiest way to do this is to replace the XSLT included with SharePoint with the XSLT you see listed here. Make sure to save a backup copy of the original XSLT from SharePoint, since you will want to reapply it after you are done retrieving the XML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="/">
    <xsl:copy-of select="*"/>
  </xsl:template>
</xsl:stylesheet>
```

Once you have applied the new XSLT, you will need to view the source on the web part zone in your browser, since the results that appear in the search results’ web part are just text. The XML for the search results is in the HTML source. Search for `<all_results>`, and you will find the XML for your search results.

**Stepping through the XSLT**

Now that you have seen the XML returned by the search engine, let’s take a look at the different parts of the default XSLT file included with SharePoint that transform this XSLT. You can completely change the XSLT file to meet your needs, but in most cases you probably will tweak what is already provided by SharePoint.

**Root Template**

The first section of the XSLT is the root template. The root template is actually at the bottom of the XSLT file and just instructs the XSL transformation to match the entire XML results. This section of the XSLT works with the parameters section described in the next section. The parameters section contains both of the web part properties that you set, such as whether to display discovered definitions. Also, this section
Chapter 9: Customizing the Search Experience

contains the more results text and link, and the parameters generated by the search engine, such as whether there were no results.

This part of the transformation looks to see whether there are keywords passed as the query or whether there were search results. If there were no results or keywords, it displays the correct error messages to the user. You will want to modify the different sections of the XSLT to customize the messages returned, such as telling the users how better to perform searches in your company’s environment, or scolding them for not putting in search keywords. If there are results, the \texttt{xsl:otherwise} section of the code is called, and the search results template is called in the XSLT.

\[
\text{<!-- XSL transformation starts here -->} \\
\text{<xsl:template match="/">} \\
\text{<xsl:if test="$AlertMeLink">} \\
\text{<input type="hidden" name="P_Query" / >} \\
\text{<input type="hidden" name="P_LastNotificationTime" / >} \\
\text{</xsl:if>} \\
\text{<xsl:choose>} \\
\text{<xsl:when test="$IsNoKeyword = 'True'">} \\
\text{<xsl:call-template name="dvt_1.noKeyword"/>} \\
\text{</xsl:when>} \\
\text{<xsl:when test="$ShowMessage = 'True'">} \\
\text{<xsl:call-template name="dvt_1.empty"/>} \\
\text{</xsl:when>} \\
\text{<xsl:otherwise>} \\
\text{<xsl:call-template name="dvt_1.body"/>} \\
\text{</xsl:otherwise>} \\
\text{</xsl:choose>} \\
\text{</xsl:template>}
\]

\textbf{Parameter Section}

As described above, the parameter section transforms the web part parameters and any parameters that the search engine passes back in the XML. If a parameter is not passed, then the transformation will perform different actions. The following is the XSLT for the parameter section:

\[
\text{<xsl:param name="ResultsBy"/>} \\
\text{<xsl:param name="ViewByUrl"/>} \\
\text{<xsl:param name="ViewByValue"/>} \\
\text{<xsl:param name="IsNoKeyword"/>} \\
\text{<xsl:param name="IsFixedQuery"/>} \\
\text{<xsl:param name="ShowActionLinks"/>} \\
\text{<xsl:param name="MoreResultsText"/>} \\
\text{<xsl:param name="MoreResultsLink"/>} \\
\text{<xsl:param name="CollapsingStatusLink"/>} \\
\text{<xsl:param name="CollapseDuplicatesText"/>} \\
\text{<xsl:param name="AlertMeLink"/>} \\
\text{<xsl:param name="AlertMeText"/>} \\
\text{<xsl:param name="SrchRSSText"/>} \\
\text{<xsl:param name="SrchRSSLink"/>} \\
\text{<xsl:param name="ShowMessage"/>} \\
\text{<xsl:param name="IsThisListScope" select="True"/>} \\
\text{<xsl:param name="DisplayDiscoveredDefinition" select="True"/>}
\]

\textbf{Parameter Section}

As described above, the parameter section transforms the web part parameters and any parameters that the search engine passes back in the XML. If a parameter is not passed, then the transformation will perform different actions. The following is the XSLT for the parameter section:

\[
\text{<xsl:param name="ResultsBy"/>} \\
\text{<xsl:param name="ViewByUrl"/>} \\
\text{<xsl:param name="ViewByValue"/>} \\
\text{<xsl:param name="IsNoKeyword"/>} \\
\text{<xsl:param name="IsFixedQuery"/>} \\
\text{<xsl:param name="ShowActionLinks"/>} \\
\text{<xsl:param name="MoreResultsText"/>} \\
\text{<xsl:param name="MoreResultsLink"/>} \\
\text{<xsl:param name="CollapsingStatusLink"/>} \\
\text{<xsl:param name="CollapseDuplicatesText"/>} \\
\text{<xsl:param name="AlertMeLink"/>} \\
\text{<xsl:param name="AlertMeText"/>} \\
\text{<xsl:param name="SrchRSSText"/>} \\
\text{<xsl:param name="SrchRSSLink"/>} \\
\text{<xsl:param name="ShowMessage"/>} \\
\text{<xsl:param name="IsThisListScope" select="True"/>} \\
\text{<xsl:param name="DisplayDiscoveredDefinition" select="True"/>}
\]

\textbf{Parameter Section}

As described above, the parameter section transforms the web part parameters and any parameters that the search engine passes back in the XML. If a parameter is not passed, then the transformation will perform different actions. The following is the XSLT for the parameter section:
Chapter 9: Customizing the Search Experience

No Search Results

The No Search Results template contains the transformation for when the search result comes back empty, either because there were no keywords entered, or the keywords entered produced no result. You can modify the text in either case to customize it to fit in your company’s environment. For example, you could link to the Search best practices’ document for your company, or to the company’s helpdesk, if folks need further help working with search. The XSLT for this section follows, and no, the misspelling in the comment at the beginning is not our doing; it is in the code!

<!-- When there is keywory to issue the search -->
<xsl:template name="dvt_1.noKeyword">
  <span class="srch-description">
    <xsl:choose>
      <xsl:when test="$IsFixedQuery">
        Please set the ‘Fixed Query’ property for the webpart.
      </xsl:when>
      <xsl:otherwise>
        Enter one or more words to search for in the search box.
      </xsl:otherwise>
    </xsl:choose>
  </span>
</xsl:template>

<!-- When empty result set is returned from search -->
<xsl:template name="dvt_1.empty">
  <div class="srch-sort">
    <xsl:if test="$AlertMeLink and $ShowActionLinks">
      <span class="srch-alertme" > <a href="{$AlertMeLink}" id="CSR_AM1" title="{$AlertMeText}" > <img style="vertical-align: middle;" src="/_layouts/images/bell.gif" alt="" border="0"/> <xsl:text disable-output-escaping="yes"> &amp;nbsp; </xsl:text> <xsl:value-of select="$AlertMeText" />&gt;</a>
    </span>
  </xsl:if>
  <xsl:if test="string-length($SrchRSSLink) &gt; 0 and $ShowActionLinks" >
    <xsl:if test="$AlertMeLink">
      |<a type="application/rss+xml" href="{$SrchRSSLink}" title="{$SrchRSSSText}" id="SRCHRSSL"><img style="vertical-align: middle;" border="0" src="/_layouts/images/rss.gif" alt=""/> <xsl:text disable-output-escaping="yes"> &amp;nbsp; </xsl:text> <xsl:value-of select="$SrchRSSSText" /></a>
    </xsl:if>
  </xsl:if>
  <br/>
  <br/>
  <span class="srch-description" id="CSR_NO_RESULTS">
    No results matching your search were found.
  </span>
  <ol>
    <li>Check your spelling. Are the words in your query spelled correctly?</li>
  </ol>
</div>
</xsl:template>
Chapter 9: Customizing the Search Experience

- Try using synonyms. Maybe what you're looking for uses slightly different words.
- Make your search more general. Try more general terms in place of specific ones.
- Try your search in a different scope. Different scopes can have different results.

Search Results
This is where all the magic happens, since the Search Results template is called if there are search results that need to be transformed. You will find that this template displays the search header and the search results, using the default properties, such as the icon, title, description, and the default metadata (path, size, author, and lastmodified). You can modify this template, as you will see later in the chapter, to customize the properties returned, and how they are displayed. This template calls other templates, such as the Hit Highlighting template.

Hit Highlighting Template
The Hit Highlighting template displays the search terms in a different color in the returned text. By default, SharePoint will bold all the matched search terms, but you can easily modify this to make the search terms all appear in the same color or different colors. We will customize hit highlighting later in the chapter. You will see in the code that follows, the XML result for hit highlighting in the returned search XML, and, below that, part of the XSLT that bolds the search terms.

Display Size Template
The Display Size template calculates the right size in which to display the result. It determines whether it should be bytes, KB, or MB, using math in XSLT, and then displays the right result customized to the size of each result returned in the XML.
Chapter 9: Customizing the Search Experience

**Display String Template**
The Display String template just displays strings from the XML, such as the author or lastmodified properties.

**Document Collapsing Template**
The Document Collapsing template is used if you have the remove duplicates enabled for your search. This template determines whether there are duplicates, and whether you have shown or removed them in the UI, and then provides either the ability to expand them or the ability to collapse them, according to the current collapsing status.

**More Results for Fixed Query**
The More Results for Fixed Query template displays a More Results link if there are more results that the user can view for a fixed query.

**Discovered Definitions**
SharePoint supports the autodiscovery of definition for terms from the search corpus, using natural language. For example, if many documents use the acronym ADFS and define it as Active Directory Federation Services, when SharePoint crawls those documents, it takes note of those definitions and can display what people are saying about that particular term. Unfortunately, this feature is not customizable nor can it be administered, so you may get some definitions, or you may not. If you do get definitions, the XSLT transforms and displays the definitions.

**Working with SPD to Create Your XSLT**
Writing XSLT by hand is not the most fun experience you will have in your development activities. To make it easier, you can use any version of Visual Studio, but it is recommended that you use Visual Studio 2005 and above or SPD to write your XSLT, rather than just using Notepad. SPD has one advantage over the other options: you can use the Data View web part to design the layout and the look and feel of your search results, and then just grab the XSLT it generates and apply the XSLT in your Core Search Results web part. Let’s take a look at the steps involved in order to do this.

First, open up a SharePoint site in SPD. This will allow you to use the Data View web part later on.

Next, you will want to grab the XML of a search result, since you will be using that in SPD as your search result to transform. The steps to do this were discussed earlier in the chapter.

Third, create a new ASPX page in SPD, and place the Data View web part on the page by going to the Data View menu and selecting Insert Data View.
Chapter 9: Customizing the Search Experience

Fourth, the datasource library pane will open, which allows you to specify the datasources to which you want to connect. Expand the XML files node, and add a new XML file. Find your search results XML file, and add it as a datasource. If asked, say OK to importing it into SharePoint. Figure 9-7 shows SPD after importing the XML file.

![Figure 9-7: Imported XML search results in SPD](image)

Next, click the drop-down on your imported XML file, and select the Show Data option to make sure that you can see the results. From there, you can select the columns that you want to include in your search results by using multi-select, by holding down the Control key, and selecting columns. Drag and drop those columns into your Data View web part. Figure 9-8 shows the default columns and how they appear in the web part.
Once you have the data in your data view, you can start using all the features of SPD to customize the layout, such as changing the main layout, changing the conditional formatting, changing any calculations or formulas, or making any other enhancements you want. Once you are done with your enhancements, you can just grab the XSLT that the SPD has generated for you from the split or code view in SPD. Find the `<xsl:stylesheet>` tag and copy everything through the closing tag for the XSL stylesheet. Figure 9-9 shows simply customized SPD search results. The XSLT transforms the image link to a picture, and uses the URL property to hyperlink the title property and displays in a single column view.
Once you have the XSLT, you can apply it to your Core Search Results web part, using the XSLT property. Once you apply the settings, you will get a result like that shown in Figure 9-10, which uses the new XSLT you created in SPD for your search results.

In this quick example, you have seen how you can use SPD to create your XSLT. SPD is just one tool that you can use. Once you get the XML file that SharePoint returns to you, you can use any designer, such as Visual Studio, to build your XSLT and apply it to SharePoint.
Chapter 9: Customizing the Search Experience

Customizing Hit Highlighting in Search

Let’s start looking at some of the most common scenarios in which you will want to customize the XSLT. This will be a good walkthrough of the XSLT structure discussed earlier in the chapter. The first scenario you will look at is customizing hit highlighting. It is one of the easiest changes that you can make that will have considerable visual effect. As you saw earlier, the XSLT for transforming the search results is broken into sections. To change hit highlighting, you need to modify the `Hithighlighting` section in the XSLT. The easiest way to find it is to search for `Hithighlighting`. The section should look like the code that follows:

![Figure 9-10: SPD XSLT applied to search results](image)

Customizing Hit Highlighting in Search

Let’s start looking at some of the most common scenarios in which you will want to customize the XSLT. This will be a good walkthrough of the XSLT structure discussed earlier in the chapter. The first scenario you will look at is customizing hit highlighting. It is one of the easiest changes that you can make that will have considerable visual effect. As you saw earlier, the XSLT for transforming the search results is broken into sections. To change hit highlighting, you need to modify the `Hithighlighting` section in the XSLT. The easiest way to find it is to search for `Hithighlighting`. The section should look like the code that follows:
Chapter 9: Customizing the Search Experience

Each of the matches that you see in the code that matches a `c` and then a number, such as 0, 1, 2 and so on, represents a search term match in the document. To change the styling for each term as it appears in your search results, you need to modify the style of each matched term to whatever you want it to be. In this example, for dramatic effect, we will modify the color, font, and size, so that you can see the changes. The modified code follows, and Figure 9-11 shows the changes in the user interface.

```xml
<xsl:template match="ddd">
   &amp;#8230;
</xsl:template>
<xsl:template match="c0">
   <b><xsl:value-of select="."/></b>
</xsl:template>
<xsl:template match="c1">
   <b><xsl:value-of select="."/></b>
</xsl:template>
<xsl:template match="c2">
   <b><xsl:value-of select="."/></b>
</xsl:template>
<xsl:template match="c3">
   <b><xsl:value-of select="."/></b>
</xsl:template>
<xsl:template match="c4">
   <b><xsl:value-of select="."/></b>
</xsl:template>
```

```xml
<xsl:template match="ddd">
   &amp;#8230;
</xsl:template>
<xsl:template match="c0">
   <strong style="color: green; font-family: arial; font-size: 10px;"> <xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c1">
   <strong style="color: blue; font-family: serif; font-size: 12px;"> <xsl:value-of select="."/></strong>
</xsl:template>
<xsl:template match="c2">
   <strong style="color: red; font-family: arial; font-size: 8px;"> <xsl:value-of select="."/></strong>
</xsl:template>
```
Adding and Rendering Custom Properties

There are times when you want to allow searches and display results of custom properties that the index engine finds in your content. By default, SharePoint does not search the custom properties that it finds until you map those properties to managed properties in the index. Once you map the properties, you need to modify the Core Results web part to retrieve both the property and the XSLT to display the property. Let’s go through the steps involved in making this work, since this will be a common operation, especially if you customize the columns in SharePoint.

First, we need to crawl whatever content sources contain the custom properties. If you can, do a full crawl, but incremental crawls should be able to pick up new properties. The index engine will find all the new properties in the content and make those part of the crawled property list. You then have to map the crawled property to a managed property in SharePoint. You can think of a managed property as a way to group properties for both administration and displaying to the end user. Sometimes you will get a number of properties that have the same meaning, since end users may create metadata in their lists, libraries, or external content sources that have different names. For example, you may find in your crawled properties Product Name, Prod Name, Product Number, or Product ID, and they may all map to the same concept as a unique identifier for a product. Rather than displaying all these different metadata elements to your users, you may want to map them all to a single managed property, so that users can search for the product they want but not have to worry about the property name or managing the property lifecycle.
Chapter 9: Customizing the Search Experience

To manage your properties, go into Central Administration and administer Search under your SSP administration pages. In the Search Settings page, you will find a link called Metadata property mappings. This is the administration page that you will use to manage your metadata mappings. Figure 9-12 shows this page.

![Metadata Property Mapping administration page](image)

**Figure 9-12: Metadata Property Mapping administration page**

In the page, you will see all your managed properties. From here, you can administer your properties. You can add new properties, remove existing properties, change the mapping, and allow a property to be used in scopes.

You can also see all your crawled properties by clicking on the Crawled Properties link on the left side of the screen. From this page, you can see all your crawl properties by category, such as Business Data, SharePoint, Web, or Office. You can discover whether your custom properties are being crawled and extracted, by searching or looking through the categories. Figure 9-13 shows the Crawled Properties user interface.
You can also modify the default settings for your crawled property categories. For example, you may want to generate a new managed property automatically for each crawl property that is discovered. This is off by default, and I’ll explain why that is a good choice in a second. You can also turn off autodiscovery or automatic mapping of string values to managed properties. One thing to realize is that SharePoint will do automatic mapping for strings. For other datatypes, you will have to manually map the crawled property to the managed property. Figure 9-14 shows the Category administration page where you can change the bulk crawled property setting.
Why would you not just allow the creation of new managed properties for every discovered crawled property? There are two good reasons: First, the naming of the crawled properties are sometimes not people-friendly. For example, if you add custom columns to SharePoint, the crawled property for those custom columns will be prefixed with `ows_` and will be encoded, so customer name becomes `ows_customer_x0020_name`. That’s not the most friendly property name for a user to select in the user interface when he or she is searching.

Second, you may want to map multiple crawled properties to a single managed property to improve relevance and the user experience. Rather than having five different author properties, you can have a single author managed property that maps to all five crawled properties.

Beyond seeing your crawled properties in this section, you can edit your crawled properties. Editing your properties lets you see the category, property set, datatype, and other relevant information about your crawled property. You can also see examples of documents that use your property and your mappings, and you can decide whether you want your crawled property to be included in the index. This last option is important to understand, since it will affect the size of your index. If you enable the inclusion of the property in the index, SharePoint will store the value for the property in the search index. This will allow searches to find not only content that matches the search term, but also anything that matches the value in the property. For example, if you enable the author scenario described above with multiple properties mapping to a single managed property, searching on a name will match content
in the documents and other content types, as well as matching values in the properties. Figure 9-15 shows the administration page for managing your crawled properties individually.

![Image](c09.indd 241)

Figure 9-15: Managing crawled properties

**Adding a New Managed Property**

To add a new managed property, click the New managed property button. In the dialog, type in a name for your managed property. You will want to remember this name, since you will be using it in an XML property in your Core Search Results web part. Also, select the datatype for your new managed property. You have a choice of whether you want to map your managed property to a single crawled property that is discovered, in priority order, or whether you want to map it to all discovered crawled properties. Most often you will want to include all crawled properties.

Click the Add Mapping button, and you will be presented with the user interface in Figure 9-16. From there, you can select the category of the crawled property by which to filter the list, or type in a property name on which to search. The screenshot shows you an example from the Business Data category (BDC), which is all the BDC crawled properties. You can select multiple properties from multiple categories to perform your mapping, but the one pet peeve is that you need to open the dialog box multiple times, since there is no multi-select capability.
The other option you have is to allow the property to be used in scopes. This is useful if the property you are mapping would make a good way to group items, using a scope. For example, you could allow a mapping by geography to be a scope, so you could scope queries to country from the BDC.

Once you are done with your mapping, perform an incremental crawl so that data associated with your mapping can be stored in the search index, if you selected that option in your configuration settings.

One quick note about site columns and crawled property mapping: You will not see your site columns appear in your crawled properties until the site column has some data in it. When you create your mapping to the site column, you will want to use the crawled property with the prefix ows_.

Figure 9-17 shows a completed configuration page for adding a new managed property.
Chapter 9: Customizing the Search Experience

Customizing Core and Advanced Search Results

Once you have completed your mappings and incremental crawls, you will want to go into your search page to modify your Core Search Results web part. You will also want to go to the Advanced Search page to modify the Advanced Search Results web part, as well. By default, both these web parts do not include custom properties that you map. You need to modify the property definition XML in order to have your property retrieved, so that you can use the property in your XSLT. Alternately, you can have the user pick your property from the drop-down list in Advanced Search.

Let’s start with modifying the Core Search Results web part. If you open the web part properties for this web part, you will see, under the Results Display/Views section, a Select Columns property. This property contains all the columns that the search engine will pass back to you, so if you want to use your new managed property, you will need to add it to the XML in this property. It’s simply a matter of adding a new Column node to the XML with the name of your property. Figure 9-18 shows the editing of the XML for this property to include a new managed property.

Figure 9-17: New Managed Property dialog box
Once you have added your column, modify your XSLT to display your new property as you want it to appear. Make sure to put the column name in all lowercase in your XSLT, or you will not get a match to it.

If you want to enable searching via Advanced Search, the steps are a little bit different. Go to your Advanced Search page and modify the Advanced Search web part. In that web part, you will find, under the Properties section, a Properties property (that’s a lot of properties in a sentence!). Edit the property definitions in the XML to add your new property definition. You will need to add it to two locations in the XML. The first is the property definition section, and the second is the property reference section for the result types. We’ll spend some time going through these two different sections.

The first section is the property definition. This section contains all the definitions for the search properties for Advanced Search that the user can select. You will need to add a node for your property to the XML; it should contain the name of the property that you defined in the managed properties configuration, the datatype of the property, and the display name for your property for the user. The code for the property definition is shown here:

```xml
<PropertyDefs>
  <PropertyDef Name="Path" DataType="text" DisplayName="URL"/>
  <PropertyDef Name="Size" DataType="integer" DisplayName="Size"/>
  <PropertyDef Name="Write" DataType="datetime" DisplayName="Last Modified Date"/>
  <PropertyDef Name="FileName" DataType="text" DisplayName="Name"/>
  <PropertyDef Name="Description" DataType="text" DisplayName="Description"/>
  <PropertyDef Name="Title" DataType="text" DisplayName="Title"/>
  <PropertyDef Name="Author" DataType="text" DisplayName="Author"/>
  <PropertyDef Name="DocSubject" DataType="text" DisplayName="Subject"/>
  <PropertyDef Name="DocKeywords" DataType="text" DisplayName="Keywords"/>
  <PropertyDef Name="DocComments" DataType="text" DisplayName="Comments"/>
  <PropertyDef Name="Manager" DataType="text" DisplayName="Manager"/>
  <PropertyDef Name="Company" DataType="text" DisplayName="Company"/>
  <PropertyDef Name="Created" DataType="datetime" DisplayName="Created Date"/>
  <PropertyDef Name="CreatedBy" DataType="text" DisplayName="Created By"/>
  <PropertyDef Name="ModifiedBy" DataType="text" DisplayName="Last Modified By"/>
  <PropertyDef Name="GlobalAuthor" DataType="text" DisplayName="Global Author"/>
</PropertyDefs>
```

Figure 9-18: Editing the Select Columns XML list
Chapter 9: Customizing the Search Experience

The property reference section, which is contained in the result type section, is where you get your property to display the drop-down list from which the user picks the property. It is also where the user types in criteria to filter on that property. You will notice that there can be multiple result types, such as All Results, Presentations, Word Documents, Excel Documents, and others. You can even add new result types, if you want to customize the search experience to limit it to search only web pages or any criteria that you specify. The result type definition is made up of a query definition and the property reference definitions. For the query, you specify the default filter. For example, for the Excel Documents result type, the query is FileExtension="xls" or FileExtension="xlsx" or FileExtension="xlt" so that the query only returns Excel documents.

In our example, we’re just going to add a new property reference in the All Results section, so that users can pick the custom property and filter on it when they search all of the results in their queries. A property reference is just an XML node with a name attribute, where that name attribute is the name of your property. The code below shows the All Results results type with the new property reference added:

```xml
<ResultTypes>
  <ResultType DisplayName="All Results" Name="default">
    <Query/>
    <PropertyRef Name="Author"/>
    <PropertyRef Name="Description"/>
    <PropertyRef Name="FileName"/>
    <PropertyRef Name="Size"/>
    <PropertyRef Name="Path"/>
    <PropertyRef Name="Created"/>
    <PropertyRef Name="Write"/>
    <PropertyRef Name="CreatedBy"/>
    <PropertyRef Name="ModifiedBy"/>
    <PropertyRef Name="GlobalAuthor"/>
  </ResultType>
</ResultTypes>
```

Once you have added definitions, your property will appear in the Advanced Search page. As you can see in Figure 9-19, the new property appears in the property picker, so users can search on that property.

![Advanced Search Property Picker](image)

Figure 9-19: Custom property in the property picker
Chapter 9: Customizing the Search Experience

Customizing People Search and Results with Custom Properties

Just as you can customize unstructured content searches, you can customize people search with custom properties. Unlike going into the crawled properties and creating a managed property, to add to people search you need to add a new property to the profile store. The new property can be mapped to an existing connection that you have for importing properties, such as Active Directory, LDAP, or the BDC, or it can be a standalone user profile property that is not imported from another source but may be entered manually by the user. Figure 9-20 shows the User Profile Properties page, where you can view, edit, or create new user profile properties.

Once you have created your new profile property, you will want to go to the People Search page and modify the People Search Box web part properties. Under the Miscellaneous section, you will find a properties property. Open that property as shown in Figure 9-21, and you will see an XML definition for all the properties for the people search.
You will notice that the property definitions for people are different from the unstructured data property definitions. The definitions point to user profile properties, rather than managed search properties. Add your new user profile property to the definition. The code that follows shows an example of a new property definition for a user profile property called \textit{geography}:

\begin{verbatim}
<Property Name="Geography" ManagedName="Geography" ProfileURI="urn:schemas-microsoft-com:sharepoint:portal:profile:Geography"/>
\end{verbatim}

Once you have added your new property and exited the edit mode, you will see your new property appear in the criteria for the advanced people search, as shown in Figure 9-22.
Chapter 9: Customizing the Search Experience

One other thing to note is that people search uses a different XSLT in its Core Results web part from the unstructured search. Therefore, you can modify the people search to include additional properties and user interface customizations to make people search more relevant to your organization. The first step to customizing people search is to change the selected columns that are returned as part of your queries. This is similar to the way you customize the unstructured search columns, in that you need to modify the people search Core Results web part and the Selected Columns property, which you will find under the Results Query Options section. Make sure that the columns you add to this XML file are Search managed properties; otherwise, you will not get any results from the additional properties. Also, if you want to retrieve values, make sure that, for the crawled property, you tell SharePoint to include the value in the index. Finally, if they are from the user profile store, make sure that you check the Indexed box for the user profile property; otherwise, the search engine won’t retrieve and store the values for the property. You will have to run another people import and perform a full crawl if you check the index box for your property, since the search engine has to retrieve the data for your newly indexed people property. I have found that sometimes you need to reset your crawled content to get some of the people properties to work. Definitely use the XML retrieval steps for your search results, discussed earlier in the chapter, to see whether your property is returned, and whether it has a value, before modifying the XSLT.
Chapter 9: Customizing the Search Experience

The code that follows shows customizing this property to add a new property. In this case, we will add the name of the manager to the selected columns list:

```xml
<Column Name="Manager"/>
```

Once we have the column available, we can modify the XSLT for the core people search results web part. There are two areas that you need to modify if you want to display a profile property, such as manager, in the sample. First, there is a section called prsch-Description, which calls the DisplayOfficeProfile template. You need to modify this section to include your property as one of the parameters sent to this template, as shown here:

```xml
<div class="psrch-Description">
  <xsl:call-template name="DisplayOfficeProfile">
    <xsl:with-param name="title" select="jobtitle" />
    <xsl:with-param name="dep" select="department" />
    <xsl:with-param name="phone" select="workphone" />
    <xsl:with-param name="office" select="officenumber" />
    <xsl:with-param name="manager" select="manager" />
  </xsl:call-template>
</div>
```

Next, you need to modify the DisplayOfficeProfile template, so that your property is included in the parameter list, and also to add the rendering of your property to the output. We modified the default display that SharePoint outputs for people search to try to make the user interface easier to read, but the concepts are completely applicable to the out-of-the-box XSLT rendering code. The code is shown here:

```xml
<xsl:template name="DisplayOfficeProfile">
  <xsl:param name="title" />
  <xsl:param name="dep" />
  <xsl:param name="phone" />
  <xsl:param name="office" />
  <xsl:param name="manager" />

  <span class="psrch-Metadata">
    <xsl:if test='string-length($manager) &gt; 0'>
      Manager: <xsl:text> </xsl:text><xsl:value-of select="$manager" /><br/>
    </xsl:if>
    <xsl:if test='string-length($title) &gt; 0'>
      Title: <xsl:text> </xsl:text><xsl:value-of select="$title" /><br/>
    </xsl:if>
    <xsl:if test='string-length($dep) &gt; 0'>
      Department: <xsl:text> </xsl:text><xsl:value-of select="$dep" /><br/>
    </xsl:if>
  </span>
</xsl:template>
```
Chapter 9: Customizing the Search Experience

Figure 9-23 shows the new people search results, which now display the manager of the person in the results as part of his or her default people properties.

Figure 9-23: Adding the Manager property to your people search results
Chapter 9: Customizing the Search Experience

**Using Fixed Queries**

One interesting customization that you can do is to use the fixed query capabilities of SharePoint and the Core Results web part or People Core Results Web Part anywhere in your SharePoint environment. For example, on your home page, you may want to have a display that shows all the new Microsoft Word documents with a certain subject added in the last 24 hours. Using the Fixed Keyword Query property on the Results web parts, you can customize the query that is performed and make it too many for users to modify your query by dragging and dropping the Results web parts on your page. You just need to set both the Fixed Keyword Query and also the Cross-Web Part Query ID properties to something besides their default values, such as the keywords and restrictions you want to use for the keyword property and something besides User query. You could also modify the XSLT to change how the search results are displayed. Using fixed queries is an easy way to display cross-site collection data in your SharePoint sites, when you combine customizing the user interface with creating custom scopes to limit your queries to the other site collections you want to aggregate.

**Adding Custom Web Parts**

When working with Search, you can also add your own web parts to the search page. How you write those web parts depends on what you want to achieve, since many of the search web parts included with SharePoint are sealed. This means that you cannot inherit from those web parts. Instead, you have to write your own web part to serve the same functionality. Also, since the results are not shared with custom web parts, you will need to grab the search terms from the querystring and write your own code to the search API in order to get the results. Some great example web parts are available as part of the Search Community Toolkit on Codeplex at [www.codeplex.com/sct](http://www.codeplex.com/sct).

The good thing about inheriting from the Core Results web part is that you do not have to rewrite all the navigation and query mechanisms, since the control already understands how to perform these functions. The bad part about using this method is you have no control over the search query that is passed. Instead, you can only modify properties on the inherited control and display the final results. The query and results are actually stored in something called srho, which is the search request hidden object. Also, the Core Results web part is the only web part from which you can inherit. All the rest you can encapsulate, but you cannot directly inherit from.

One useful example of inheritance is retrieving the XML of the results by using the xmlResponseDoc property of the Core Results web part. Then, you can perform your own visualization of the results.

The more common scenario will be that you will write your own web parts that will sit on the page with the Microsoft web parts. The issue with this solution is that you will have to perform your own queries using the search APIs to get back your results. This means that you will have to retrieve the query from the querystring for the page, and then spin up your own objects to perform the query. It is not difficult, but it is cumbersome. You can retrieve the querystring in your web part by using the context object, which contains the request object, which has a collection called querystring. In that collection, you will want to retrieve the search term by using “k” as your index. Sample code is:

```csharp
this.Context.Request.QueryString["k"]
```

Your other option for customizing the search experience is to gather the search parameters in a custom application, parse the parameters, and then pass the parameters along to the default search experience in SharePoint via a URL.
Chapter 9: Customizing the Search Experience

Search Community Toolkit

One set of tools that I want to point out, since they are useful community supported tools, is the Search Community Toolkit on Codeplex. You can find it at www.codeplex.com/sct. In the toolkit you will find a faceted navigation web part, search as you type, a really useful Search Service tool that we will discuss in the API chapter 12; ASP.NET search controls; and a number of other useful tools. If you have not downloaded the toolkit, definitely take a look, since source code is included with all the tools.

Conclusion

In this chapter, you have seen how you can customize the Search user experience in a number of different ways, from simple customizations using properties to more complex customizations using XSLT and custom web parts. If you are going to get more complex in your customizations, you will want to learn the Search API and XSLT in depth, since you will be spending a lot of time writing to the API and transforming the results that you get back.
Understanding and Tuning Relevance

There are multiple factors that contribute to users’ satisfaction with a search engine. These include precision (they find the right answers), recall (they find all the answers), search site layout and branding (how easy it is to use), and the speed with which results are returned. The most important factor, however, is relevance.

What Is Relevance?

Relevance is your success metric as a Search administrator, it’s the thing that your users will love you for or hate you for. It can make you or break you. OK, maybe that’s starting to get a little melodramatic, but relevance really can’t be overstated when it comes to user satisfaction of a search engine. If the results don’t give a user what they were looking for, then what’s the point?

Before we get any further into this chapter, it’s probably best to get a couple of definitions out of the way. For a lot of people, relevance and ranking are one and the same, and the terms are used interchangeably. This isn’t actually true.

Ranking is the mathematical process of ordering a set of results in a specific order, based on the result of a set of algorithms that take many different parameters about the content into account.

Relevance is the subjective user happiness with that rank. It’s a subtle difference, but it’s an important difference. You can make relevance better without changing the ranking algorithms, by changing several factors, such as the user experience and the way that content is stored, or by using thesaurus and noise word lists. In fact, there are many things you can do to make a huge difference, and you’ll never go near a ranking parameter, a property weight, or a saturation constant.
Chapter 10: Understanding and Tuning Relevance

This chapter will cover many of the options available to you to optimize the built-in ranking parameters and elements, and also how to change some of the ranking parameters directly, to enable you to achieve your desired end result.

Our parting shot, before diving into the details, is that there’s no magic bullet. No single change will likely yield perfect results. Any changes you make in an effort to improve relevance should be done through an iterative process. Change too much in one go, and you’ll have no idea whether one change was great and the others were bad. The end result would be to have evened things out, such that things look unchanged.

Built-In Ranking Elements and Algorithms

Ranking elements are the inputs into the ranking algorithms, so understanding the ranking elements SharePoint Server 2007 and Microsoft Search Server 2008 use out-of-the-box will not only help you optimize the configuration options available in the UI and through the object model, but also give you some insight into how some of the ranking elements are used internally to enable you to optimize your content.

SharePoint Server 2007 and Search Server 2008 include two types of ranking elements, Dynamic and Static. Dynamic elements vary depending on the query terms a user submits, such as keyword, property, or URL matching. Static elements are those that are query independent and are tied to the content, rather than the query, and include elements, such as file type biasing, URL depth, and Click Distance.

Both SharePoint Server 2007 and Search Server 2008 include the same ranking algorithms, which are configured identically. They use several ranking algorithms that include a modified version of the Okapi BM25 algorithm, as well as algorithms that process the static ranking elements and high confidence matching for exact keyword matches and best bets. A combination of the results of all of the algorithms involved is what determines the final rank of a result. Okapi BM25 is based on the probabilistic retrieval framework developed by Stephen E. Robertson, Karen Spärck Jones, and others. There are several sites on the Internet that have very good explanations of how the algorithm works (including the Wikipedia).

A major difference between the current release of SharePoint Server 2007 and the previous version, SharePoint Portal Server 2003, is that there is now a default AND of query terms, rather than the OR (or sometimes AND) in SharePoint Portal Server 2003. This change alone has led to a substantial increase in relevance between the product versions.

Understanding and Tuning Relevance

There are two approaches to improving or tuning the relevance of the results your users see when they run queries. This section will walk through the options within each approach.

The first approach is to tune and tweak the factors and elements that are used by the ranking algorithms; this is an indirect way to improve relevance. You’re using the built-in features, and by understanding how things work, you can optimize your configuration and content in a way that bumps the relevance of results in a positive direction without changing any of the math behind the ranking algorithms.
Chapter 10: Understanding and Tuning Relevance

The second approach is to change some of the ranking parameters which, in turn, changes the way that the ranking algorithms work and obviously affects the order of the results you’ll get back. This chapter is split into two sections.

1. Things you can change outside of the ranking algorithms
2. Things you can change that directly affect the ranking algorithms.

Using either approach, or a combination of both approaches, should help you improve recall and precision of results based on your corpus and requirements.

As mentioned at the start of this chapter, the key to a successful tuning exercise is to go through many small iterative changes, rather than lots of changes in one go. Through a repeating cycle of planning, making a change, and testing, you’ll achieve positive results much faster than trying to do everything at once.

There are several very thorough white papers and documents on MSDN and TechNet on how to plan, structure, and build a comprehensive relevance test harness, that include the benefits and limitations of side-by-side comparisons, and a guide to create a very thorough statistical evaluation process.

So, in the interests of space (and not wanting to regurgitate some great content that’s already published and publically available), I’m not going to recycle that content, but I’d absolutely recommend that you read all you can about how to evaluate relevance before you dip your toes any further in the water. A good starting point is this article from MSDN at http://msdn.microsoft.com/en-us/library/bb499682.aspx

**Things You Can Change outside of the Ranking Algorithms**

We’ll start with the things that you can change outside of changing the numerical weights and constants that plug into the ranking algorithms. There are a lot of factors in and around the content that you can tweak and change to influence the relevance of the results.

**Content Management**

It might sound very obvious, but when it comes to Search, content is king. To deliver a high-quality search experience to your users will take a little (to a lot of) work. Just setting the crawler loose, and leaving it for a few days to follow all the links it finds, will certainly stuff a lot of content into the index, but if that content will be the content that your users want to find will be somewhat of a lottery.

So, this section will cover how to get the right content into your index in the first place, how to get the wrong content out of your index if it finds its way in, and how to guide users into the right behavior for finding what they want. This section will also cover some details on how the indexer interprets URLs and links it finds in web pages and other indexed content.
Chapter 10: Understanding and Tuning Relevance

Content Sources

The first thing you’ll probably do while configuring search is set up several content sources and check the Start full crawl box. This approach is great for a demo, but not so great for your users in the long run.

Before you add any content sources, try spending a few hours pulling an inventory together of the content repositories that are useful for your organization, and asking others what repositories are important to them. It will be time well spent. If you already have usage data for internal web sites, then try mining that to help identify what sources are going to be valuable. There’s usually no shortage of voices when you start asking these sorts of questions, and the information you gather is generally well worth the effort.

Getting the right content into the index in the first place is important for achieving great relevance. It’s slap-on-the-forehead obvious, but it’s surprising how often it’s overlooked, even after people start to complain that the results they’re getting aren’t relevant, or that results from sources that users would expect results from are simply, completely absent.

It’s rarely a very complicated thing to figure out what data are important to your users, but it can sometimes take some leg work and some time. Get this bit right, and you’ll be a long way towards being set up for success.

Once you’ve got your content repositories identified and created, you’ll need to experiment with a couple of Content Source configuration parameters, namely the Crawl Schedules and the Crawl Settings.

These two settings can have a positive or negative effect on relevance, depending on how you configure them, and what your corpus looks like in terms of content churn (changed content) and “noise content” (bad content that can demote good content or dilute the result set).

Crawl schedules ultimately control the freshness of the index, as the index contains a snapshot of what the content looked like at the moment it was crawled. How you configure your crawl schedules is going to come down to more than just keeping indexed content fresh and results relevant and accurate — crawling content puts a load on the index server and SQL database and can slow users’ query response time — so, it’s a balancing act between keeping your index fresh and keeping the servers running efficiently. There’s no right or wrong answer. You must determine, based on your organization’s requirements and your server hardware, the optimal Crawl Schedule settings for your content sources, but keep in mind that an out-of-date index generally leads to a drop in relevance.

Crawl settings control the depth to which the crawler will follow links when crawling your content (see Figure 10-1). They are a great way to help reduce the amount of unwanted noise content in your index. Crawl settings are covered in detail in Chapter 4, and using them cleverly can make a very positive impact on overall relevance.
Chapter 10: Understanding and Tuning Relevance

Figure 10-1: Screenshot of crawl settings
Chapter 10: Understanding and Tuning Relevance

The final two configuration settings covered in this section are Crawl Rules and Search Result Removal; again, both of these settings are covered in Chapter 4, but it’s worth a very quick review in this section. They can both have a positive effect on relevance and are your Swiss Army knife for removing lots of noise content from your index.

Using a crawl rule can prevent everything from a specific document to whole web site, file share, or any other content repository from being indexed. This means that if you have content sources configured that you actually want to index, but they contain links to other sites that you don’t want to index (as well as links to ones that you do want), or even content within them that you don’t want, you can configure a crawl rule to exclude specific content as it’s crawled. This helps to keep the noise out and the good stuff in.

If you already have noise content in your index that you want to remove, you can use the Search Result Removal feature. Any site, document, or content type you remove with this feature will immediately prevent the content from appearing in search results. In addition, it will create a new crawl rule that will prevent the content from being reindexed.

Fine-tuning the content in your index can be tricky, so remember to use information sources like the crawl logs, and reporting and filtering features to help you make the right decisions when setting up new content sources, and modifying and tweaking existing content. We’ve found it useful in the past to set up a test server and do a dry run of a new content source before configuring it on a production server. You can thereby check to see what sort of content you’re getting back in isolation from everything else; it’s a real timesaver and helps you to identify crawl rules and crawl settings in advance.

Scopes

Custom scopes are a handy way to filter results behind the scenes with simple or complex filter rules. Creating scopes is covered in detail in Chapter 4, and apart from the convenience factor they provide, they’re a very useful tool for presenting smaller, more relevant, and targeted result sets to your users without the users having to do anything particularly hard when they submit their queries.

To the end user, any configured scopes appear in a drop-down list box next to the search box, or are transparently tied to a search tab. The users can quickly filter their searches for a specific configured topic. Search tabs are only supported in SharePoint Server 2007; they are not supported in Search Server 2008.

Again, there’s no right or wrong answer as to what scopes you should create, or how you should configure the filters. This should be driven by your users’ general behavior. In general, keep the number of scopes low. Pick the top three to five content areas, by looking at query log reporting or using some other method, and build your scopes to target those areas.

Creating, managing, and monitoring scopes can be a very effective way of driving your end users’ search behavior in the right direction and helping them to find the right result, which means better relevance for them.

Metadata Management

Metadata management is a tough problem. There have been many books written on the topic, so we won’t delve into the details in these few paragraphs. We do want to highlight a few key areas that can make a big difference when we’re talking about metadata, search, and relevance.

Several metadata fields are very important to search. Getting accurate and descriptive data into them can make a huge difference to a document’s discovery, later on.
Chapter 10: Understanding and Tuning Relevance

One of the most important metadata fields is the title field of a document; this applies to any document — a web page, a Word document, a PDF document, or any other type of document that has a title. Finding a way to encourage your users to use descriptive titles will pay dividends.

The examples that follow highlight the good and bad ways to name a document or web page:

1. “Home page – Corporate Finance and Accounting” rather than “Home page - FINWEB”
3. “Microsoft Search Server 2008 Overview and What’s New.pptx” rather than “MSSOaWN.pptx”

Another very helpful and related end user behavior is to use spaces in filenames, including web pages. Using spaces allows the word breakers to do their job properly on the filename when the content is indexed, and allows them to extract words from the filename, rather than a long string of text.

Following are some good and bad examples:


In addition to the content title, the content author metadata is of high value to Search. Often, you know that person X authored a document that you want to find about topic Y, but there could be a huge amount of content about topic Y, making it difficult to find that exact document you’re looking for, unless you can filter the results by author. Again, it pays dividends to encourage your users to ensure that the author property contains the name of the person who authored the content.

There are three metadata properties that will, by default, directly affect the ranking of results (i.e., they are mathematically weighted to boost their relevance). They are Title, Author, and File Name. It’s possible to add more properties to this list, which is explained later in this chapter, but, out-of-the-box, those three will repay you with the largest reward in relevance improvement, if you spend some time optimizing them.

If you decide to take a look at sharpening up your metadata, there’s no need to stop at just Title, Author, and File Name. Every metadata field on every document that you get accurate data into is going to help Search, particularly on documents that don’t contain a lot of text, such as spreadsheets, images, and image-rich documents.

**Anchor Text**

Anchor text is extracted from anchor tags, `<A ... /A>`, which are found in the HTML of web pages and other documents that include hyperlinks. The anchor text is usually a very strong indicator of the content that the URL points to and can heavily influence relevance.

---

*The %20 in the URL is a single space that has been URL encoded. The filename for the page in the filesystem would be ..\Expense Reporting Rules.aspx.*
Chapter 10: Understanding and Tuning Relevance

If you have a lot of web content or web accessible content (e.g., file shares exposed through IIS), then it’s a good idea to review how your content is linked on the pages that you consider authoritative within your organization.

For example, the text and link below will achieve the same results for users browsing the page but will be interpreted very differently by a search engine:

To find out more about paternity leave please visit the HR Intranet Site by clicking here.

The anchor text would be `<A href=http://hrweb>clicking here</A>`, which is perfectly fine for a human user, but doesn’t help a search engine much to understand the context of the URL. A more search engine (and relevance) friendly way of linking to the HR intranet site would be:

To find out more about paternity leave please visit the HR intranet site.

The anchor text in this case would be `<A href=http://hrweb>HR Intranet Site</A>`, which is much more useful to a search engine in associating the two things together.

Search Server 2008 and SharePoint Server will index the anchor text from the following sources:

- HTML anchor elements
- Microsoft Windows SharePoint Services link lists
- Microsoft Office SharePoint Portal Server 2003 listings
- Hyperlinks found in Microsoft Office Word 2007, Microsoft Office Excel 2007, and Microsoft Office PowerPoint 2007 files that are saved in the OpenXML format
- Any other file types handled by third-party IFilters that emit hyperlinks

URL Depth

The URL depth-ranking element measures the number of `/`'s in a URL and computes a value that will boost the ranking of sites that have fewer `/`'s in the URL.

In general, content on sites that have shallow URL’s will be more relevant than sites that have very deep URL’s. Organizations tend to allocate shorter and root URLs to their most important content, and this behavior acts as a reasonably strong indicator of the relevancy of content stored on a site.

Language

Language plays an enormous role in achieving relevant search results, yet its importance is often understated. In this section, we’ll walk through the key areas for which language plays a role in affecting the relevance of results and highlight some configuration features that can make a large difference in the quality and accuracy of content, when it’s indexed, and can improve the quality of results at query time.

Language Detection at Index Time

When content is indexed by Search Server 2008 or SharePoint Server, the indexer will attempt to determine the language in which the content was written, by looking for locale information extracted from each chunk of the document by the relevant IFilter. For example, all of the Microsoft Office file formats include a locale identifier.
Chapter 10: Understanding and Tuning Relevance

The most relevant language word breaker of the 49 available (see the subsection later in this section that discusses word breakers) will then be loaded to break up the textual content of the document by finding word boundaries in the text and splitting the text into tokens that are then stored in the index. Additionally, the language of the complete document will also be recorded, which can then be used to filter results by language in the Advanced Search page.

The selection of the relevant word breaker and the language classification of the whole document are controlled by two different processes. The relevant word breaker is selected per chunk of text returned by the IFilter, so that documents containing text in several languages will be word broken correctly. The overall document language is controlled by a Language Auto Detection algorithm that selects the most appropriate language based on the overall content of the document, not per chunk.

If the indexer can’t identify the language of the content, it will default to using the word breaker that matches the system locale of the server running the indexing role. For example, if you have the French language version of Windows Server installed, and the indexer can’t find any locale information in a document, the document will be indexed using the French word breaker by default. It’s also worth noting that it’s the SKU installation language that matters, not any language packs you install afterwards, so be sure to keep that in mind when installing the operating system on your index server.

Search Server 2008 and SharePoint Server can auto-detect and word break content written in the languages in Table 10-1.

<table>
<thead>
<tr>
<th>Arabic</th>
<th>Hindi</th>
<th>Punjabi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bengali</td>
<td>Hungarian</td>
<td>Romanian</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>Icelandic</td>
<td>Russian</td>
</tr>
<tr>
<td>Catalan</td>
<td>Indonesian</td>
<td>Serbian</td>
</tr>
<tr>
<td>Chinese</td>
<td>Italian</td>
<td>Slovak</td>
</tr>
<tr>
<td>Croatian</td>
<td>Japanese</td>
<td>Slovenian</td>
</tr>
<tr>
<td>Czech</td>
<td>Kannada</td>
<td>Spanish</td>
</tr>
<tr>
<td>Danish</td>
<td>Korean</td>
<td>Swedish</td>
</tr>
<tr>
<td>Dutch</td>
<td>Latvian</td>
<td>Tamil</td>
</tr>
<tr>
<td>English</td>
<td>Lithuanian</td>
<td>Telugu</td>
</tr>
<tr>
<td>Finnish</td>
<td>Malay</td>
<td>Thai</td>
</tr>
<tr>
<td>French</td>
<td>Malayalam</td>
<td>Turkish</td>
</tr>
<tr>
<td>German</td>
<td>Marathi</td>
<td>Ukrainian</td>
</tr>
<tr>
<td>Greek</td>
<td>Norwegian</td>
<td>Urdu</td>
</tr>
<tr>
<td>Gujarati</td>
<td>Polish</td>
<td>Vietnamese</td>
</tr>
<tr>
<td>Hebrew</td>
<td>Portuguese</td>
<td></td>
</tr>
</tbody>
</table>

Table 10-1
Chapter 10: Understanding and Tuning Relevance

Language Detection at Query time

When a user submits a query, the server needs to be able to figure out what locale the user is in, so that it can word break the query terms appropriately. This is done through the browser “Accept-Language” header that is sent with the HTTP get request for the search page.

Good recall can be dependent on using the same language word breaker, for query terms, that was used to index the target content originally. Without this matchup of query terms language and indexed terms language, it can be difficult to achieve good relevance, especially when there is a large difference between the language of the word breaker and the content. For example, an English word breaker with Japanese content query terms will be interpreted differently from the terms in the index, so there’s a good chance that a query term won’t match an expected result because the words in the index have been tokenized in one language, and the query is being sent tokenized in a different language. This can cause lots of head scratching in multilingual deployments where expected results aren’t showing up and relevance seems to be less than optimal.

To help mitigate problems in a multilingual deployment, make sure that your users have the correct browser language set (this is generally the most common cause of unexpected results); guide users to the Advanced Search page, where it’s possible to select a preferred language; and ensure that your index server operating system is installed in whatever you want your default language to be, as this is the locale that will be used, if no other locale information can be found when content is indexed.

Word Breakers and Stemmers

Word breakers perform linguistic analysis on the textual content of documents as part of the indexing process, to find word boundaries in the text and to split the text into word tokens that are then stored in the index. They are used again at query time to tokenize a user’s query terms.

Stemmers are only used at query time, and, if enabled, will expand a user’s query terms to include inflectional variants of the query terms (suffixes that don’t change the words meaning). For example, if stemming were enabled for English (it’s not, by default), and a user ran a query for the word “Search,” he or she would get results returned for, not only exact matches on “Search,” but also any inflectional variant of “Search,” including “Searched,” “Searching,” and “Searches.”

Enabling stemming is only useful for languages that have significant morphological variation among their word forms. For example, it has no effect on Vietnamese, which is an analytic, or isolating, language. There are also a couple of exceptions to the rule that stemmers are only used at query time. They are also used at index time for Arabic and Hebrew content for query performance reasons, as these languages can have a very large number of forms of a single word, sometimes into the thousands. In this case, the root form of the word is used, both for the index and the query.

Stemming is enabled or disabled as a Web Part property found on the Core Results web part, as shown in Figure 10-2. To turn stemming on, simply check the box, and to turn stemming off, uncheck the box.
Enabling stemming can sometimes reduce relevance rather than improve it, and it can also increase server query load significantly, as the server is processing an increased quantity of more complex queries. Nevertheless, it's an easy thing to turn on and off, so it's worth an experiment to see what effect it has on the quality and accuracy of your results.

The rules that word breakers and stemmers use vary by language, and the word breaking and stemming architecture is extensible for SharePoint Server and Search Server 2008. You can write your own if you wish, and there are several third-party solutions to enhance support for languages that are already supported and to add support for languages that are not supported out-of-the-box.

If an unsupported language is detected in content (some locale information exists in the content, but there is no word breaker or stemmer installed to support the locale), a neutral word breaker is used that breaks apart the text at neutral characters, such as spaces and punctuation marks. If no locale information is available in the document, the word breaker and stemmer that match the index server operating system locale is used.

It's worth spending a little time explaining how stemming works, and how to enable or disable it for queries, because, depending on the content and language makeup of your corpus, it can have both a positive and negative impact on relevance.
Chapter 10: Understanding and Tuning Relevance

The languages in Table 10-2 are supported by native word breakers and stemmers in SharePoint Server 2007 and Search Server 2008. For languages marked with an asterisk (*), the stemming feature is on by default; for all other languages, it is switched off by default.

Table 10-2

| Arabic    | Hungarian* | Romanian* |
| Bengali   | Icelandic* | Russian* |
| Bulgarian* | Indonesian | Serbian_Cyrillic* |
| Catalan   | Italian    | Serbian_Latin* |
| Croatian  | Japanese   | Slovak*   |
| Czech*    | Kannada*   | Slovenian* |
| Danish    | Korean     | Spanish*  |
| Danish    | Latvian*   | Swedish   |
| Dutch     | Lithuanian*| Tamil*    |
| English   | Malay      | Telugu*   |
| Finnish*  | Malayalam* | Thai      |
| French*   | Marathi    | Turkish*  |
| German*   | Norwegian_Bokmaal | Ukrainian* |
| Greek*    | Polish*    | Urdu*     |
| Gujarati  | Portuguese | Vietnamese |
| Hebrew    | Portuguese_Brazilian | |
| Hindi     | Punjabi    |           |

Noise Words

Noise words are those words that do not add any value to search and are ignored both at index and query time. The list of noise words is configurable through a text file, by language, stored in the following directory, for both SharePoint Server 2007 and Search Server 2008:

```
..\Program Files\Microsoft Office Servers\12.0\Data\Applications\<GUID>\Config
```

The <GUID> part of the path will be a real GUID that refers to a specific Shared Service Provider (SSP). If you have more than one SSP defined, you’ll see multiple <GUID> directories.

If you browse around and take a look at the contents of some of the directories with the \data folder from the path above, you’ll notice that the noise word files are also present in the following directory:

```
..\Program Files\Microsoft Office Servers\12.0\Data\Config
```
Chapter 10: Understanding and Tuning Relevance

The files in this folder are copied to the `<GUID>\Config` directory when a new SSP is provisioned, so, if you’re making any changes, be sure to make the change in the correct `<GUID>\Config` directory, rather than the `<Data>\Config` directory; otherwise, your changes will never be picked up (unless you create a new SSP, of course). If you have multiple query servers or a separate index server, you’ll also need to replicate manually any changes you make to one, across all of them.

Within `<GUID>\config` directory you’ll find several files named `noiseXXX.txt`, where `XXX` is the language to which the noise file applies. So, for example, the English language noise file is named `noiseeng.txt`, and the French language noise file is named `noisefra.txt`.

Opening one of these files in Notepad will allow you to edit the default entries or add custom entries. You just need to add one word per line. For reference, the `noiseeng.txt` file contains the following words, all of which will be ignored during indexing and querying:

```
a
and
is
in
it
of
the
to
```

If a noise file does not exist for a particular language, the neutral noise file, `noiseneu.txt`, is used. If you decide to make an addition or deletion in a noise file, you’ll need to perform a full crawl of your content sources for the changes to take effect.

Noise words can have a significant effect on relevance in both directions, but careful and selective use in certain circumstances can substantially improve the relevancy of your results.

**Thesaurus**

Modifying the thesaurus allows both query term expansion and replacement, and, similarly to the noise word files, there’s a thesaurus file for each language, which can be found in the following directory.

```
..\Program Files\Microsoft Office Servers\12.0\Data\Applications\<GUID>\Config
```

All of the thesaurus files are copied from the `<data>\config` directory to `<GUID>\Config` when a new SSP is created, so, if you’re making any changes to the thesaurus files on a server that already has an SSP, you’ll need to be sure that you change the files in the `<GUID>\Config` directory and not the `<data>\config` directory. If you have multiple query servers or a separate index server, you’ll also need to replicate manually any changes you make to one, across all of them.

The naming convention for the thesaurus files is `tsXXX.xml`, where `XXX` equals the language to which the thesaurus applies. For example, the English language thesaurus file is named `tseng.xml`, and the French language thesaurus file is named `tsfra.xml`.

All of the thesaurus files only contain sample content by default, and, if a thesaurus file doesn’t exist for a particular language, the neutral thesaurus, `tsneu.xml`, is used. The neutral thesaurus is also applied to every query, regardless of language, in addition to any language-specific thesaurus.
Chapter 10: Understanding and Tuning Relevance

Thesaurus files are written in XML, and as an example, the default thesaurus file for English language looks like this:

```xml
<XML ID="Microsoft Search Thesaurus">
<!-- Commented out
<thesaurus xmlns="x-schema:tsSchema.xml">
<diacritics_sensitive>0</diacritics_sensitive>
<expansion>
    <sub>Internet Explorer</sub>
    <sub>IE</sub>
    <sub>IE5</sub>
</expansion>
<replacement>
    <pat>NT5</pat>
    <pat>W2K</pat>
    <sub>Windows 2000</sub>
</replacement>
<expansion>
    <sub>run</sub>
    <sub>jog</sub>
</expansion>
</thesaurus>
-->
</XML>
```

An expansion set is a group of words that are synonyms of each other. If a query term matches one of the expansion set members, then the query is expanded to include all of the other members. For example, if you wanted a query for “House” to include the terms “Condo” and “Apartment,” you would add an expansion set within the relevant language thesaurus file and between the `<thesaurus>` tags that looks like this:

```xml
<expansion>
    <sub>House</sub>
    <sub>Condo</sub>
    <sub>Apartment</sub>
</expansion>
```

It’s also not possible to have duplicate entries in your expansion groups as the example below illustrates. In this case only the first expansion set configured in the list will ever be used for a query containing the term “House.”

```xml
<expansion>
    <sub>House</sub>
    <sub>Condo</sub>
    <sub>Apartment</sub>
</expansion>

<expansion>
    <sub>House</sub>
    <sub>Live</sub>
    <sub>Reside</sub>
</expansion>
```
A replacement set specifies a pattern that, if found in a query, will be replaced by one or more other terms. For example, you could create a replacement set for the terms “MSS” or “MSS2008,” which contains “Microsoft Search Server.” Only results that contain the words “Microsoft Search Server” are returned. No results would be returned for documents that contain “MSS” or “MSS2008.”

The example just used would look like the XML snippet that follows and would be inserted between the `<thesaurus>`...`</thesaurus>` tags, with the `<pat>`...`</pat>` tags indicating the terms you wish to match, and the `<sub>`...`</sub>` tags indicating the term you wish to use as the substitution term for a match.

```
<replacement>
  <pat>MSS</pat>
  <pat>MSS2008</pat>
  <sub>Microsoft Search Server</sub>
</replacement>
```

There are a few caveats to using replacement and expansion sets. The query terms are not treated as case sensitive, so “MSS” is the same as “mss.” All of the XML tags in the files, apart from the `<XML>` and `</XML>` tags, must be lowercase, the `<diacritics_sensitive>` tag is ignored, and it’s not recommended that you exceed 10,000 entries in a single thesaurus for query performance reasons.

Thesaurus files are only used for FREETEXT queries that are not restricted by quotation marks. For example, a query of “MSS” (with quotation marks) will not cause the thesaurus to be activated. However, a query of MSS (without quotation marks) will cause the thesaurus to be activated, and, using the preceding replacement set, results would only be returned for “Microsoft Search Server.”

You might also find a few articles on the Internet that explain how to configure the thesaurus files for SharePoint Server 2007 that include details on adding word weighting and configuring individual word stemming. Both of these configuration properties do not apply to SharePoint Server 2007 and Search Server 2008. They do, however, apply to the previous version of SharePoint, SharePoint Portal Server 2003.

The recommendations for using thesaurus file expansion and replacement sets are very similar to the recommendations for noise words, careful selective use in certain circumstances can lead to better relevance. It’s very easy to go over the top with the thesaurus and significantly reduce the quality of results, so plan and research any changes you’re thinking of making, and test them thoroughly afterwards.

**Diacritics**

A diacritic mark is a symbol that’s added to a letter to change the pronunciation or meaning of a character or group of characters; they are sometimes referred to as accent marks. The main use of a diacritic is to change the sound of a single character, but they can also be used to change the meaning of a whole word.

A specific mark may also be a diacritic in one language but not in another. For example, in Catalan, Portuguese, and Spanish, “u” and “ü” are considered to be the same character, while in German, Estonian, Hungarian, and Turkish they are considered to be separate characters.
Chapter 10: Understanding and Tuning Relevance

In SharePoint Server 2007 and Search Server 2008, diacritic support is switched off by default. To enable it, an administrator must run the following command line operation, which enables diacritic support on an SSP-by-SSP basis.

```
stsadm -o osearchdiacriticsensitive -ssp <ssp name> [-setstatus <True|False>] [-noreset] [-force]
```

Depending on the primary language of your users, enabling diacritics may enhance recall, that is, the number of results returned, but it may also decrease relevance because you have more results. It may also have the opposite effect and improve relevance. It completely depends on the languages you operate with, and the effectiveness of enabling diacritic support will vary from organization to organization.

**Custom Dictionaries**

Custom dictionaries can be used to identify words that you want a word breaker to ignore. They’re essentially word breaker exclusion lists, and can be useful if you have words and acronyms in your corpus that might otherwise be broken into separate tokens or ignored completely, but are actually important for you to have in your index.

There are no custom dictionaries installed by default. You’ll need to create them in the following directory on all of your query servers, as well as your index server:

```
..\Program Files\Microsoft Office Servers\12.0\Bin\
```

You’ll need to create one file for each language for which you need to implement a custom dictionary and follow the naming convention of `customXXX.lex`, where `XXX` is the language identifier. Custom dictionaries use the same naming convention as noise and thesaurus files for the language identifier. For example, a custom dictionary for English would be named `customeng.lex`, and for French it would be `customfra.lex`.

The files should be text files in Unicode format and contain one word per line, separated by a carriage return/line feed, and shouldn’t contain more than 10,000 entries.

With the exception of the characters pipe (|), white space, and hash (#), any alphanumeric characters, as well as punctuation and normally breaking characters and other symbols, can be entered in the custom dictionary. Pipe and white space cannot be used anywhere in a word entered in the custom dictionary, and hash cannot be used at the beginning of a word in the custom dictionary, but it can be used within the word or at the end of the word.

Entries consisting of multiple words on a line separated by white space are not supported, and entries must consist of single words only and are not case sensitive. Entries in the custom dictionary do not get normalized, nor do they get alternative outputs; they are treated as exact matches only, and the maximum length of an entry is 128 (Unicode) characters.

After adding or editing an existing custom dictionary file, you’ll need to restart the search service for the change to take effect.
Understanding Users’ Query Behavior with Query Logs

SharePoint Server 2007 and Search Server 2008 both include a set of built-in search usage reports. The reports are grouped into two groups, one for queries and one for results. The reports are accessible from two places, the first is through the Search Administration dashboard in Central Administration (the usage reports group in the left navigation pane), and the second through the site settings page on a Search Center site (in the Site Collection Administration group, Search Queries and Search Results).

The information contained in the various reports can be a gold mine for anyone wanting to provide an optimal search experience. See Table 10-3.

In the remainder of this section, we’ll go through the individual reports, and highlight how you could use report information to improve both the end user search experience and other aspects of your intranet design.

Table 10-3

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries over Previous 30 Days, and Queries</td>
<td>Both of these reports provide trending data, which is useful for figuring out how many queries your servers are answering over time for planning purposes, but they’re not particularly useful for improving relevance.</td>
</tr>
<tr>
<td>over Previous 12 Months</td>
<td></td>
</tr>
<tr>
<td>Top Query Origin Site Collections over Previous 30 Days</td>
<td>If you’re running SharePoint Server 2007 in farm mode and have several site collections that consume search services from the SSP you’re looking at the report from, then this report cab be very informative in giving you an idea of which site collections are driving search traffic. Again, this information is very useful but not necessarily in terms of Search.</td>
</tr>
<tr>
<td>Queries per Scope over Previous 30 Days</td>
<td>The Queries per Scope report can be useful for improving the end user search experience. It will validate or refute the decisions you made when creating your custom search scopes. If after you’ve created several custom scopes you still see a very high proportion of queries using the default scope, then there’s a good chance that either you created the wrong scopes and no one’s using them, or, users don’t know how to use them. Either way, this should set you off in the right direction — to get your scopes set up correctly or to educate your users.</td>
</tr>
<tr>
<td>Top Queries over Previous 30 Days</td>
<td>The queries you see in this report are what users are looking for most frequently. It gives you the guidance to do things, like add best bets and definitions, and more importantly, it allows you to check that you’ve got the right content sources being indexed in the first place. If you start seeing queries for content that you know isn’t even being indexed, you can take remedial action quickly.</td>
</tr>
<tr>
<td>Search Results Top Destination Pages</td>
<td>This report helps you understand what results users are clicking through to, and can help you build the right navigational elements on your intranet sites. If you find that a lot of users are hitting a certain page through search results, then it might be more user-friendly to create a direct link to that page from the navigation on your intranet.</td>
</tr>
</tbody>
</table>

(Continued)
Chapter 10: Understanding and Tuning Relevance

Table 10-3 (Continued)

<table>
<thead>
<tr>
<th>Report Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries with Zero Results</td>
<td>The content you see in this report is about as big a red flag as you can get that something is not right. This could be the case for any number of reasons, including content not being indexed, end user error, admin error, or that the content simply doesn’t exist, and thus can’t be indexed. Whatever the reason, this report will give you a good pulse on users’ satisfaction with Search.</td>
</tr>
<tr>
<td>Most Clicked Best Bets</td>
<td>This report can be used in a similar way to the Search Results Top Destination Pages report, in that it can give you clues that people are using search to find content that might be better linked directly to navigation or prominent site content.</td>
</tr>
<tr>
<td>Queries with No Best Bets</td>
<td>This is a great report for enhancing the end user search experience by giving users what they want. If there are a lot of queries for a particular topic, and no best bet exists for the query terms, then it’s going to pop out in this report, and you can use the info to build an indirectly user-generated collection of best bets.</td>
</tr>
<tr>
<td>Queries with Low Clickthrough</td>
<td>A set of results that don’t get a clickthrough implies that the users didn’t find what they were looking for. You can analyze this data to figure out if relevant results are actually being returned, but for some reason the UI is making it difficult for users to identify them; or that the results really are poor quality, and you can take measures to try and improve them.</td>
</tr>
</tbody>
</table>

**User Interface**

The user interface for the Search Center in SharePoint Server 2007 and Search Server 2008 is very customizable. Changes made to it can have a profound effect on users’ perceived relevance of results. Testing has shown that differences in the presentation style, layout, and branding can yield double-digit variance in perceived relevance, when the result sets are exactly the same.

Just about all of the options and features described in this section have little or no effect on your actual data, the index, or the ranking algorithms, but they do have a direct effect on users’ happiness, which unconsciously affects how they think about the results they’re getting, and how they can navigate through them. There are various options on just about every Search web part to control behavior that directly affects the user experience. In this section, we’ll cover some of the most common settings you can change to optimize the way that results are displayed.

**The Search Box**

In organizations of all sizes, there will always be some people that know how to construct a good query, and some who don’t. Generally, those who do will find what they’re looking for faster than those who don’t and will have a much better search experience.

It pays to train your users about how to search properly. It’s very effective to add some text to the search page, usually in a dynamic HTML flyout, next to the search box that explains how to construct a query that includes phrase matching with “quotation marks” and property:value filtering. There are many tech-savvy people who don’t know about or don’t use either of those syntaxes when they search.

Table 10-4 is a quick recap of the “advanced” keyword query syntax. If you teach your users how to construct an advanced query, your perceived relevance will take an immediate jump for the better.

270
Chapter 10: Understanding and Tuning Relevance

Table 10-4

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Character</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple term</td>
<td>Keyword with no special requirements</td>
<td>N/A</td>
<td>Vegetable</td>
</tr>
<tr>
<td>Phrase</td>
<td>Match of all keywords in quotes</td>
<td>&quot;...&quot;</td>
<td>&quot;Growing turnips&quot;</td>
</tr>
<tr>
<td>Included term</td>
<td>Keyword that must be in content items returned in search results</td>
<td>&quot;+&quot;</td>
<td>+carrot</td>
</tr>
<tr>
<td>Excluded term</td>
<td>Keyword that must not be in content items returned in search results</td>
<td>&quot;-&quot;</td>
<td>-cabbage</td>
</tr>
<tr>
<td>Property Filter</td>
<td>Property specified must exist on the document for the document to be returned in search results</td>
<td>PropertyName: value</td>
<td>genus:allium</td>
</tr>
<tr>
<td>Property Filter Prefix</td>
<td>Property prefix wildcard must match for the document to be returned in search results</td>
<td>PropertyName:val*</td>
<td>genus:alli*</td>
</tr>
</tbody>
</table>

Custom XSL for Hit Highlighting

Highlighting the right information in a set of results in a way that users are comfortable with, will help them navigate the result sets much more efficiently. Two of the ways you can achieve this are: to customize the XSL in the Core Results web part to highlight information that might be more useful than just the query terms the user searched for (which is the default), or to change completely the look and feel of the results or individual results based on the type of result.

For example, you could have customers displayed one way, resellers another, and internal web sites in a third format. You’ll need to be careful not to go overboard with the customization, or you’ll end up with an early ‘90s animated-GIF-type web site, but a little customization can go a long way. Chapter 9 covers customizing XSL in detail.

Duplicate Collapsing

With duplicate collapsing switched on, results that are found to be very similar are collapsed into a single result when displayed to a user. The documents don’t have to be identical, and can contain small differences and still be considered duplicates. The near-duplicate algorithm used was developed by Microsoft Research, and the process is called “Shingling.” There are no parameters available to tune this process, but, overall, it works extremely well.

Duplicate collapsing is turned on by default and is a configuration parameter on the Core Results web part. When results are returned that are considered duplicates, they appear as a single result in the result set with the addition of a View Duplicates link below, which, if clicked, will display all of the identified duplicate documents.

Using duplicate collapsing can greatly enhance relevance because it avoids the “page full of the same document” experience if many copies of the same document have been indexed from different locations, which is a common occurrence in many organizations. Instead of 20 identical results showing up, the user is presented with only one result. This cleans up the result set allowing other relevant or related documents to make it onto the first page, too.
Chapter 10: Understanding and Tuning Relevance

Even though this feature is switched on by default, it’s worth mentioning, as there may be occasions when it makes sense to switch it off. The good news, though, is that, as it’s configured on the Core Results web part, it’s easy to switch on or off, and if you need to switch it off, you can do so without affecting any other Core Results web parts.

Advanced Search

The Advanced Search page allows you to build a custom search experience for your users that exposes more advanced query options than are available through the keyword syntax, and also provides a UI to make it easier for users to build complex queries.

There are several articles on the Microsoft TechNet and MSDN sites that detail how to customize and enhance the Advanced Search page, and include directions on how to add extra properties, languages, and result types to allow you to create a page that suits the requirements of your users. For example, http://msdn.microsoft.com/en-us/library/bb428648.aspx is a great step-by-step example of how to add additional properties to the Advanced Search page.

Using the Advanced Search page can be preferable to the regular search page in certain scenarios (see Figure 10-3). It is generally most useful in custom search application deployments such as Call Center and Manufacturing applications, and when users need to find documents in specific languages.

Figure 10-3: Advanced Search
Chapter 10: Understanding and Tuning Relevance

Automatic Definitions

Automatic Definitions is a feature that’s only available in English, that will detect phrase and definition pairs in indexed content, usually matching acronyms with their full definition. The feature works by looking for cue phrases in content, such as “is the,” “was the,” and “is called,” and by inspecting the word prior to the cue phrase and those words after the cue phrase.

It is not possible to edit, prepopulate or change any of the automatically discovered definitions. Automatic Definitions is enabled and disabled through the Core Results web part property Display Discovered Definition in the Results Display/Views group.

Automatic definitions are not highly useful in the context of relevance, but they’re worth pointing out, as Automatic Definitions is an often unused feature, and in some scenarios it can provide value by finding something that may have otherwise been missed.

Did you mean? Query Suggestions

The “Did you mean?” feature provides alternative query terms for queries that are identified as being misspelled. Suggestions for alternative terms are drawn from the query logs, a static custom dictionary, and a dynamic custom dictionary built from words found in the index.

The Did you mean feature? is switched on by default and shows up in the Search Summary web part. The feature is not configurable, you can’t edit, add, or remove suggestions, and it’s only available in the English, French, German, and Spanish languages.

Using Search Center Tabs

Search Center tabs are available in SharePoint Server 2007 but not Search Server 2008. A tab can be tied to a specific scope, so the tabs provide a very handy way to guide users visually into using the right scopes for their query.

Tabs are easy to set up through the UI (check the built-in help for instructions), and if you’ve decided to configure more than a handful of scopes, they’re a great way to make the most used ones easier for users to get to (you can find out which are used the most, through the built-in Queries Per Scope over Previous 30 Days report). See Figure 10-4.

Figure 10-4: Search Center tabs

Again, the use of tabs is a “less is more” exercise. Careful consideration of what tabs you need to create, and monitoring of which scopes are being used most frequently, will help you make the right decisions in creating tabs that are used, rather than ignored.

High-Confidence Matches

High-confidence matches are a way to promote certain results or content into a prominent position in the result set or somewhere else on the results page, when a query term returns an exact match of a configured set of properties, a defined set of keywords and synonyms, or an exact match to a URL.
Chapter 10: Understanding and Tuning Relevance

We’ll walk through the three types of high-confidence matches in this section, and cover how each one works, and how you can configure them to work more effectively for your corpus.

Keywords and Best Bets

Best bets are editorial content, and links that are defined and scoped at the Search Center level under the Site Settings, Search Keywords UI. Best bets allow you to define editorial results for common query terms by specifying the trigger Keyword Phrase and Synonyms. If a user’s query matches a best bet keyword phrase, or one of the synonyms, then the best bet result is displayed in the Best Bet web part, along with any results found in the index and any federated results, if any federated locations have been configured on the page.

Figure 10-5 shows the Add Keyword dialog, where you can create a new best bet that includes the keyword phrase, as well as any synonyms you want to add. It’s also possible to set a publishing window for best bets, so they only show up between specific dates.
Best bets don’t directly influence the relevance of results, but they do provide a huge amount of value for users if the best bets are created in a structured way and kept up to date. Using the Most Clicked Best Bets and the Queries with No Best Bets reports will help you keep your best bet content up to date, accurate, and, most importantly, relevant. Never underestimate the power of human-generated content in the form of a best bets for keeping your users buzzing along (see Figure 10-6).

Figure 10-6: Example of best bets and definition for the query term “SharePoint”

Best bets can be displayed in either the Best Bets web part or the High Confidence Results web part (see Figure 10-7). Both the Best Bets and the High Confidence web parts are actually the same web part; their default configuration is just set up slightly differently. High-confidence results are discussed in the next section.

Figure 10-7: Default configuration settings for a Best Bet web part
Chapter 10: Understanding and Tuning Relevance

Best bets are available in both SharePoint Server 2007 and Search Server 2008 and behave in exactly the same way in both products. They’re a very powerful way to increase user satisfaction with Search and should be high up on your list of things to configure, sooner rather than later.

High-Confidence Results

You can think of high-confidence results as something along the lines of automatic best bets. If a query term exactly matches a value for a predefined property, it’s considered a high-confidence match, and the first result (by default) is displayed in a High Confidence Result web part. This allows the high-confidence result to be pushed to somewhere prominent on the page to catch a user’s attention. Figure 10-8 shows a high-confidence result.

![Figure 10-8: A high-confidence result](image)

As described in the previous section, high-confidence results can also be displayed in a Best Bets web part, and best bets can also be displayed in a High Confidence Results web part — the two web parts are interchangeable, and it’s just a matter of setting a few settings on either web part to configure them to display both types of results, or only one type. Figure 10-9 shows the default configuration settings for a High Confidence web part. (Notice that the options are identical to those shown in Figure 10-7 for the Best Bets web part, just the selected check boxes are configured differently).

A high-confidence result is returned if a query term exactly matches one of the managed properties mapped to the property HighConfidenceMatching. By default, the properties that are mapped to HighConfidenceMatching are:

- People:PreferredName
- People:WorkEmail
- People:UserName
- People:AccountName

If, for example, a user searched for “rriley,” and that was my Active Directory account name, there would be an exact match against the People:UserName property. This would trigger a high-confidence result.
You’re free to add any properties you wish to this list or to edit the default ones. It only really makes sense to add properties to this list that are either unique (for example, a property that maps to part numbers, or a property that maps to account codes), or that you’ll get a very small set of hits against.

It’s not going to help you much if you add a property here with values that could be repeated multiple times, like the Author property. You will only see the first result displayed in the web part, so you’ll lose pretty much all of the high-confidence bit, as you’ll see the first result of possibly hundreds or thousands, as opposed to the only result, or the first of a handful. While it is possible to increase the number of high-confidence results displayed through the web part configuration settings, doing so can counter the point of having a high-confidence result in the first place.
Chapter 10: Understanding and Tuning Relevance

We’ll explain how you can customize properties that are returned with the high-confidence results. Customization is driven through managed properties. If you take a look at the metadata properties list (through the Metadata Properties link in the left navigation pane in the Administration dashboard), and scroll about halfway down the page, you will see a set of managed properties called HighConfidenceDisplayProperty1 to HighConfidenceDisplayProperty15, as shown in Figure 10-10.

![Figure 10-10: High-confidence display properties in the Metadata Property Mappings UI](image)

Many of these properties are mapped to people properties by default, but you can edit, add, or remove any of the properties mapped to them. For example, if you had a property called PartNumber that you mapped to HighConfidenceMatching to trigger a high-confidence result, and you also had a PartDescription property that you wanted returned with any results, you could simply map PartDescription to one of the HighConfidenceDisplayProperty properties, and it would render on the results page through a High Confidence Results web part.

There’s one extra step you can take, after you’ve mapped your properties to HighConfidenceMatching to trigger a match, and then mapped all the other display properties you need to one or more of the...
Chapter 10: Understanding and Tuning Relevance

HighConfidenceDisplayProperty properties, and that’s to configure the XSL on the web part that you’re using to display the results.

If you look back at Figure 10-10, you can see that there’s an XSL Editor . . . button. You can modify or replace the default XSL to render your high-confidence results to suit your requirements.

Now, there are a couple of caveats worth covering at this point. The first, is that high-confidence results are configured by default to work only for people results in SharePoint Server 2007. The second is that, if you have Search Server 2008, then all of the features are there, but they’ll never be triggered, because Search Server 2008 doesn’t include, or have the ability to index, a profile store as SharePoint Server 2007 does, which is where all the people information comes from.

Search Server 2008 still has all of the managed properties you need, and they’re actually all configured exactly the same as in SharePoint Server 2007 and pointing to the same People properties that will never contain data. If you want to use high-confidence results with Search Server 2008, you’ll probably want to strip all of the preset property mappings out and write a new XSL. The one that ships is also the same one that ships with SharePoint Server 2007 and is geared towards displaying people data.

With SharePoint Server 2007, what ships as default works very well for people results, as it was configured to do in the first place. Enhancing people results by possibly adding extra people-related properties is quite an easy task; just map the new property to one of the HighConfidenceDisplayProperty properties, and you should be set. The default XSL should just take care of displaying the new property. Configuring high-confidence results for other types of content is a little trickier. It probably involves writing a new XSL and adding a secondary High Confidence Result web part to the page, so that you don’t interfere with existing people results.

The final thing to be aware of is that the built in XSL renders properties from HighConfidenceDisplayProperty1 to HighConfidenceDisplayProperty11, but it won’t display any values from HighConfidenceDisplayProperty12 to HighConfidenceDisplayProperty15. You’ll need to edit the XSL to add logic to render these properties in the web part. We mention this because 12–15 have no properties mapped to them by default and are a prime target for the first place to which you would add your new custom property. Unless you’ve also edited the XSL on your web part, then your changes aren’t going to show up. If you want to leave the XSL alone, I’d suggest picking one of the HighConfidenceDisplayProperty properties below 11 to map your custom property to.

High-confidence results are an underused tool in the relevance tweakers tool belt. They’re a little bit involved to get your head around, if you’re not comfortable with managed properties or XSL, but once you figure out how they work, they can pay you back in spades by automatically pushing certain types of information front-and-center for your users. I’d thoroughly recommend kicking the high-confidence results tires and spending some time figuring out how they work; chances are they’ll be able to add value.

URL Matching

The URL of a web site is very often a good indicator of the site’s purpose, and when sites are created, the owners usually go to a fair amount of work to make the URL guessable and descriptive.

If a user’s query matches the display name or site name of the site, then the result will be considered a high-confidence match and displayed as the first result within the Core Results web part. The user’s query terms are also concatenated with the spaces removed to check for a match against a URL, and if a match is found, it’s returned as the first result.
Chapter 10: Understanding and Tuning Relevance

For example, a query of “Windows Media” would match http://windowsmedia as a high-confidence match, and a link to http://windowsmedia would be the first result returned.

Naming your intranet sites concisely and consistently will greatly aid in later discovery through Search.

Things You Can Change That Directly Affect the Ranking Algorithm

Everything discussed so far in this chapter has revolved around how to increase perceived relevance without directly touching the ranking algorithms.

If you follow the guidance in this chapter, you won’t have changed any of the math that’s going on behind the scenes, but you may have tweaked your content. Perhaps the built-in ranking settings work better for your users, or maybe you changed the look and feel of your search site, or maybe you just understand why some things work the way they work and have changed a couple of configuration settings that make them work better for you. Whatever you’ve done so far, you’ve left the math that goes into ranking at arm’s length.

In some cases, usually in very specific use cases, it’s beneficial to get under the Search hood and directly influence the ranking algorithms by changing some of the values they use to compute rank. This section will walk through the options available to you.

Before we get into the detail, we need to mention a “relevance” health warning. The values that are set by default in SharePoint Server 2007 and Search Server 2008 (the ranking parameters are configured identically in both products) are the result of a very significant amount of detailed testing, retesting, and testing again. If you decide that you need to change any of the parameter values described in this section, then do so with great care and a lot of validation, both before you make the change and after you make the change. It’s very easy to make a very large net negative change when you start to poke around in the math.

That being said, in addition to the brain surgery, the topics covered in this chapter include details on configuring authoritative and nonauthoritative pages through which you can directly affect the ranking algorithms by making changes though the administration pages (and not directly poking around with the math). These pages are very easy to change and can have significant value, and changing them is a thoroughly recommended practice, so that’s where we’re going to start.

Authoritative and Nonauthoritative Pages

Authoritative and nonauthoritative pages rely on a concept called click distance. Click distance is a measure of the number of clicks a piece of content is from a URL you have defined as an authoritative or nonauthoritative page.

What that translates to in terms of ranking is content that’s fewer clicks away from a configured authoritative page that has its ranking boosted, and content that’s closer in clicks to a configured non-authoritative site that has its ranking reduced.

The terms “page” and “click” in this section are slightly misleading, as is the text in the administration UI and in-box help text where these settings are configured. The term “pages” in “authoritative pages” implies that only HTTP URLs are valid input. While this is likely to be your most common type of
Chapter 10: Understanding and Tuning Relevance

authoritative “page,” it’s not the only type of URL that you’re going to have in your index. For example, you might have some file shares indexed, in which case you’d have file:// URLs; you might also have indexed content from Lotus Notes and could have some notes:// URL’s, and probably several others.

Unfortunately, the UI doesn’t make it clear that you can use any URL as an authoritative or nonauthoritative “page,” yet using any other type of URL is perfectly valid.

It’s also perfectly valid to configure the root of web site or the root of file share as an authoritative and nonauthoritative “page,” as well as a specific page on a web site or a specific document in a file share, along with any other end point in a URL.

Figure 10-11 shows the authoritative and nonauthoritative pages UI. This page is found under the Authoritative Pages link in the left navigation pane on the Administration dashboard.
Chapter 10: Understanding and Tuning Relevance

URL’s that are added to this page will have a direct effect on the ranking of content to which they link.

If you make changes to this list, you won’t see the changes reflected in search results, until either an incremental or full crawl of a content source finishes, or you click the Refresh Now check box, and click OK after you make any changes.

If you select “Refresh now,” the process will generally be very fast. For example, it should only take a few minutes to update a 30 million item index.

Configuring accurate and relevant authoritative and nonauthoritative pages can significantly improve relevance. It gives you the ability to tell Search what content is important and relevant to your organization, and what is not. This can only be a good thing for improving relevance, and it should be configured on every single installation of SharePoint Server 2007 and Search Server 2008, no matter how small or large the deployment.

**Property Weighting**

Query terms that match specific managed properties can provide strong indicators of a relevant result. Take, for example, the Title property of a document, which is derived from a document’s filename.

Document titles can be very clear indicators of the content in the document, and many document authors will spend a little time making the title of their document descriptive. There are cases, however, when content authors don’t provide descriptive titles for documents, maybe because that’s just the way they work, or because the document was based on a template, or any one of a multitude of other reasons. This leaves you with two problems: Managing content with “bad” properties, and working out which properties are important, to achieve relevant results.

SharePoint Server 2007 and Search Server 2008 provide a couple of ways to help with the problem of documents containing “bad” properties through the automatic enhancement and extraction of document metadata.

Using a built-in entity extraction algorithm, SharePoint Server 2007 and Search Server 2008 can extract the title and author for Microsoft Word, PowerPoint, and Excel files. Those properties are stored as “shadow” properties that are transparent to the user and are used in relevancy calculations in exactly the same way as the real properties. In some cases, when it’s clear that the document property is incorrect, the property is actually replaced with a more useful value extracted from the document.

This process is not configurable or pluggable (so you can’t write your own entity extractor for other properties), and it only works for Word, PowerPoint, and Excel file types, but it can substantially boost relevance for these types of documents that are stored with “bad” properties.

The problem is working out what properties are important to achieve relevant results; setting a relevance weight for those properties is configurable.

By default, the following three properties have a default assigned weight, Author (8.215), Filename (29.43), and Title (75.855). The value that can be assigned to a property ranges from zero to infinity. For example, if a user submits a query that contains a keyword or keywords that match the value of one of those three properties, the document or documents that contain those properties will have their rank boosted.
All other managed properties apart from Author, Filename, and Title have a weight of 0, by default. Properties that have a weight of 0 are not considered by the ranking algorithm, so they have no effect on relevance.

We'll cover how to set and change property weights later in this chapter, along with another configurable property relevance setting called Length Normalization.

It's worth making an important distinction at this point. Property weighting and property filtering are two different things that are not related but easily confused. Property filtering will filter a result set based on the presence of a specific property and value before the results are ranked. Property weighting is used to influence the ranking of a result set based on query terms matching the value of a specific property.

Modifying the length normalization setting applies only to properties that contain text, and it allows properties with a short amount of text to have their relevance boosted compared to a property that contains a lot of text, which will have its relevance lowered.

For example, in a scenario where relevance is calculated for two content items containing the query term within the body of the content item, one long and one short, the longer property is likely to contain more instances of the query term. Therefore, it might receive a higher rank value, even if the shorter document is just as relevant to the user's search. The length normalization setting addresses this issue, providing consistency in ranking calculations for text properties, regardless of the amount of text within that property.

The range of possible values for length normalization is between 0 and 1. If this setting is 0 for a managed property, length normalization is turned off for that property, and conversely it has the greatest influence for properties that have a setting of 1.

For long text-managed properties, this value should be set to around 0.7. For managed properties that contain a small amount of text, but that are important to relevance, use the Title managed property's value for this setting, which is 0.498.

The default weights and length normalization values for the three properties that have weights by default are shown in Table 10-5. The range value for a Property Weight is from 0 to infinity, and the range value for Length Normalization is from 0 to 1.

<table>
<thead>
<tr>
<th>Property</th>
<th>Weight Default Value</th>
<th>Length Normalization Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>8.215</td>
<td>0.63</td>
</tr>
<tr>
<td>Filename</td>
<td>29.43</td>
<td>0.736</td>
</tr>
<tr>
<td>Title</td>
<td>75.855</td>
<td>0.498</td>
</tr>
</tbody>
</table>
Chapter 10: Understanding and Tuning Relevance

It’s not possible to edit, add, or remove the weight and length normalization property values through the UI. The only way to make a change to these settings is through the Search object model. There are several examples published on MSDN on how to use the object model to change these settings, and, for your reference, the following two code snippets show how to use the object model to display all of the managed properties and their weights, along with how to display all of the length normalization values for all of your configured managed properties.

Display the weight of all the managed properties through code with C#:

```csharp
// Replace <SiteName> with the name of a site using the SSP.
string strURL = "http://<SiteName>";
Schema sspSchema = new Schema(SearchContext.GetContext(new SPSite(strURL)));
ManagedPropertyCollection properties = sspSchema.AllManagedProperties;
foreach (ManagedProperty prop in properties)
{
    Console.WriteLine("Name: " + prop.Name + "  Value: " + prop.Weight.ToString());
}
```

Display the normalization of all managed properties through code with C#:

```csharp
// Replace <SiteName> with the name of a site using the SSP.
string strURL = "http://<SiteName>";
Schema sspSchema = new Schema(SearchContext.GetContext(new SPSite(strURL)));
ManagedPropertyCollection properties = sspSchema.AllManagedProperties;
foreach (ManagedProperty prop in properties)
{
    Console.WriteLine("Name: " + prop.Name + "  Value: " + prop.LengthNormalization.ToString());
}
```

In addition to writing code to change these parameters, there are a few community-written utilities that allow you to make changes without opening Visual Studio. A search on the Internet for “SharePoint Search Relevance Tool” without the quotation marks should set you off in the right direction.

**Global Ranking Parameters**

In addition to changing the weight and/or length normalization of managed properties, or adding a weight and length normalization to a new managed property, there are several global ranking parameters that affect the rank of all results.

The various global parameters are shown in Table 10-6, along with the value range within which the parameter can be configured. Changing the values of any of these parameters will affect the rank of every result. The default settings are the outcome of many hours of testing and have been optimized for most use cases. Changing these settings can significantly degrade relevance much more readily than increase it.
## Table 10-6

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>k1</td>
<td>16.404</td>
<td>0 to infinity (normally not more than 100)</td>
<td>Saturation constant for term frequency</td>
</tr>
<tr>
<td>Kqir</td>
<td>2.12766</td>
<td>0 to infinity (normally not more than 100)</td>
<td>Saturation constant for click distance</td>
</tr>
<tr>
<td>wqir</td>
<td>36.032</td>
<td>0 to infinity</td>
<td>Weight of click distance for calculating relevance.</td>
</tr>
<tr>
<td>Kud</td>
<td>9.174312</td>
<td>0 to infinity (Normally not more than 100)</td>
<td>Saturation constant for URL depth.</td>
</tr>
<tr>
<td>wud</td>
<td>31.468</td>
<td>0 to infinity</td>
<td>Weight of URL depth</td>
</tr>
<tr>
<td>languageprior</td>
<td>0</td>
<td>Negative infinity to 0</td>
<td>Weight for ranking applied to content in a language that does not match the language of the user</td>
</tr>
<tr>
<td>filetypepriorhtml</td>
<td>166.983</td>
<td>Negative infinity to positive infinity (normally −200 to +200)</td>
<td>Weight of HTML content type for calculating relevance</td>
</tr>
<tr>
<td>filetypepriordoc</td>
<td>163.109</td>
<td>Negative infinity to positive infinity (normally −200 to +200)</td>
<td>Weight of Microsoft Office Word content type for calculating relevance</td>
</tr>
<tr>
<td>filetypepriorppt</td>
<td>163.367</td>
<td>Negative infinity to positive infinity (Normally −200 to +200)</td>
<td>Weight of Microsoft Office PowerPoint content type for calculating relevance</td>
</tr>
<tr>
<td>filetypepriorxls</td>
<td>153.097</td>
<td>Negative infinity to positive infinity (normally −200 to +200)</td>
<td>Weight of Microsoft Office Excel content type for calculating relevance</td>
</tr>
<tr>
<td>filetypepriorxml</td>
<td>158.943</td>
<td>Negative infinity to positive infinity (normally −200 to +200)</td>
<td>Weight of XML content type for calculating relevance</td>
</tr>
<tr>
<td>filetypepriortxt</td>
<td>153.051</td>
<td>Negative infinity to positive infinity (normally −200 to +200)</td>
<td>Weight of plain-text content type for calculating relevance</td>
</tr>
<tr>
<td>Filetypepriorlistitems</td>
<td>0</td>
<td>Negative infinity to positive infinity (normally −200 to +200)</td>
<td>Weight of list item content type for calculating relevance</td>
</tr>
<tr>
<td>Filetypepriormessage</td>
<td>160.76</td>
<td>Negative infinity to positive infinity (normally −200 to +200)</td>
<td>Weight of Microsoft Office Outlook email message content type for calculating relevance</td>
</tr>
</tbody>
</table>
Chapter 10: Understanding and Tuning Relevance

Some of the more accessible global parameters are those that affect the weights of several types of content that include HTML, Word documents, PowerPoint presentations, Excel spreadsheets, XML files, Text documents, SharePoint list items, and email messages.

Those are the parameters on which you should focus your energy, if your usage scenario dictates that some of these content types are more important to your users than others. The order in which they are considered relevant, based on the built-in default parameter values, from highest relevance to lowest relevance, is:

1. HTML web pages
2. PowerPoint presentations
3. Word documents
4. XML files
5. Excel spreadsheets
6. Plain-text files
7. List items

So if this order of relevance for these specific content types is giving you spurious results for your corpus, you might want to think about modifying some of them. It’s worth pointing out though, that it’s not possible to add to or remove any of the global ranking parameters.

Similarly to adding a property weight and normalization length, changes to global parameters must be done through code; you won’t find a configuration page in the administration site that lets you modify any of these values. Details on how to change these parameters are documented on MSDN, and, for your reference, code to display all of the global ranking parameters would look something like:

```csharp
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.SharePoint;
namespace DisplayRankingParameters
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                string strURL = "http://<YourSiteNameGoesHere>";
                SearchContext context;
                using (SPSite site = new SPSite("http:// YourSiteNameGoesHere "))
                {
                    context = SearchContext.GetContext(site);
                }
                Ranking sRanking = new Ranking(context);
                RankParamCollection rankParams = sRanking.RankingParameters;
                foreach (RankingParameter rankParam in rankParams)
                {
                    // Display the name and value for each parameter...
                }
            }
        }
    }
}
```
Chapter 10: Understanding and Tuning Relevance

There are a few community-based utilities that will allow you to change these parameters without cracking open Visual Studio. A search on the Internet for “SharePoint Search Relevance Tool” without the quotation marks should set you off in the right direction.

Conclusion

Hopefully, this chapter has given you insight into some of the most important aspects of modifying your content and search configuration to influence relevance, as well as how you can get your hands dirty with the ranking algorithms to make changes directly to the way rank is computed.

Whatever guidance you decide to follow from this chapter, we can’t overstress the importance of obtaining a relevance baseline before you make any changes, and then being disciplined in the changes you make, and the retesting you do after a change. Relevance is a fickle thing, and it can bite you hard if you don’t approach it with a plan and a way to prove you’ve actually done some good. If you ride in cowboy style, you’ll be chased out of town pretty quickly.
Building Applications with the Search API and Web Services

While there are many rich applications you can build out of the box with SharePoint Enterprise Search, there are times when you will want to automate, extend, or customize the search experience and services with your own code or applications. SharePoint provides a rich platform to do this, both for administrative tasks and Enterprise Search usage. The platform is provided both as a .NET object model that you can call on the server side, and as a remote API (or off-the-server machine) via standard web services interfaces. Using the platform, you can create new content sources, work with schema, perform searches, customize search results, and customize many functions that your applications require. This chapter will step you through how to leverage the platform programming pieces of Search inside of your own applications.

Getting Started with References

Before starting to write your code to the object model, you will want to add a few references to your Visual Studio projects. You will want to add the Microsoft.Office.Server reference to your project, which will allow you to get the SSP and search context. You will also want to add a reference to Microsoft.Office.Server.Search. This is where you will find the administration and programming interfaces for search. For ease of use, you will want to add namespaces, as well, in your code that defines these references.

Administration Object Model

With SharePoint, you may want to automate your administration experience or build administration tools based on your business needs. SharePoint provides an administration object model that you can call to be able to perform different operations, such as working with content
Chapter 11: Building Applications with the Search API and Web Services

sources, working with crawls, managing schema, or viewing logs and editorial results. All of these
capabilities are exposed through the Microsoft.Office.Server.Search.Administration
namespace. Please note that this namespace is not a remote, so your code will have to run on the
SharePoint Server. The rest of this section will go through the object model to show you how you can use
the object model to perform a number of different administrative functions. To help you visualize the
administration object model, Figure 11-1 shows a diagram of the object model.

Search Administrative Object Model

Figure 11-1: Search administration object model

SearchContext Object

Since all of search is implemented using the SSP model, you need to be able to get the SSP in which
search is implemented. To help with that, SharePoint provides the SearchContext object as your entry
point into working with the Search object model. This object’s GetContext method returns the context
for your site, server, or SSP. The SearchContext class has two members that you can use to get your
context. The first is the GetContext method, which is overloaded according to where your code is
running. This method takes three possible arguments. The first is a string that identifies the search
application name. This method can only be used when running in the context of a SharePoint hosted
page. The second possible argument is a reference to a ServerContext object. You would have had to
have retrieved the ServerContext object already using the SharePoint GetContext method. The final
argument is an SPSite object. SharePoint will then use the site to go find the right SearchContext to
return. One thing to note about using the SPSite argument is that you should either declare your
SPSite using the keyword in C#, or explicitly call the dispose method on the SPSite object when you
no longer need it. If you do not do this, you will find that you will have memory leaks in your code. The
reason for this is that some SharePoint objects are not correctly cleaned up by the .NET GC, so it’s best
practice to call dispose on certain objects. For more information on this, check out the following MSDN
Chapter 11: Building Applications with the Search API and Web Services

article: “Best Practices: Using Disposable Windows SharePoint Services Objects,” http://msdn.microsoft.com/en-us/library/aa973248.aspx. The following code retrieves the SearchContext using the SPSite method, where the sitename is the name as the site.

```csharp
using (SPSite site = new SPSite(sitename))
{
    context = SearchContext.GetContext(site);
}
```

Once you receive the SearchContext object, you can work with some of the global properties and methods for the object. Table 11-1 describes the interesting members of this SearchContext instance:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocationConfigurations</td>
<td>Returns a LocationConfigurationCollection object, which allows you to query or set new federated locations.</td>
</tr>
<tr>
<td>Reset</td>
<td>Resets all the data in your index but keeps the configuration. Only call this if you truly need to reset your content, since you will need to recrawl all your content.</td>
</tr>
<tr>
<td>UpdateAlertNotificationFormat</td>
<td>This method takes a string, which is the XSL stylesheet used to format your alerts. This allows you to customize your alerts based on your business needs.</td>
</tr>
<tr>
<td>UpdateDiacriticSensitive</td>
<td>Turns on or off accent sensitivity.</td>
</tr>
<tr>
<td>UpdateQueryLoggingEnabled</td>
<td>Turns on or off query logging, which in turn gives you better insight to the types of queries that users are generating, at the cost of some performance hit for logging, but not a significant amount. The performance hit is worth the user query insight.</td>
</tr>
</tbody>
</table>

**Content Sources**

One of the most common administrative operations you will do is to work with content sources in creating, modifying, or deleting them and forcing the crawl operations to work on them. The API for content sources is very straightforward once you have your SearchContext object. You will spend a lot of time with the Content and ContentSource classes which are the base classes for content sources. Depending on the type of content source you are working with, you will use different classes, such as the FileShareContentSource or the HierarchicalContentSource.

**Working with the Content Class**

Before working with content sources in SharePoint, you first need to get the Content class. From this class, you can instantiate all the other classes required to work with your content sources. To retrieve the Content class, you instantiate a new Content object and pass in your existing SearchContext object to...
the constructor. The code below shows how to perform this step. Once you instantiate the `Content` object, you can then use all of its methods and properties to access the rest of the API.

```java
Content content = new Content(SearchContext);
```

Table 11-2 lists the properties and methods of the `Content` class:

### Table 11-2

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ActiveDocuments</code></td>
<td>A string array for the active documents for a SSP.</td>
</tr>
<tr>
<td><code>ClientCertificateNames</code></td>
<td>A string array that contains the client certificates that the crawler can use.</td>
</tr>
<tr>
<td><code>ContentSources</code></td>
<td>Returns the content sources available as a collection of <code>ContentSource</code> objects.</td>
</tr>
<tr>
<td><code>CrawlMappings</code></td>
<td>Returns the crawl mappings available as a collection of <code>CrawlMapping</code> objects.</td>
</tr>
<tr>
<td><code>CrawlRules</code></td>
<td>Returns the crawl rules as a collection of <code>CrawlRule</code> objects.</td>
</tr>
<tr>
<td><code>DefaultGatheringAccount</code></td>
<td>A read only property that returns the name of the account used for crawling.</td>
</tr>
<tr>
<td><code>DeleteCrawlInProgress</code></td>
<td>Returns a Boolean indicating whether a crawl in progress can be deleted.</td>
</tr>
<tr>
<td><code>ExtensionList</code></td>
<td>Returns the included or excluded file types as a collection of <code>Extension</code> objects.</td>
</tr>
<tr>
<td><code>ListKnownLotusNotesDatabases</code></td>
<td>Returns the list of Lotus Notes databases as a string array for the server name you pass in as a string.</td>
</tr>
<tr>
<td><code>ListKnownLotusNotesServers</code></td>
<td>A string array containing defined Lotus Notes servers.</td>
</tr>
<tr>
<td><code>LotusNotesConfigured</code></td>
<td>A Boolean, which specifies whether Lotus Notes crawling is configured.</td>
</tr>
<tr>
<td><code>PauseBackgroundActivity</code></td>
<td>Pauses all content related activity.</td>
</tr>
<tr>
<td><code>ResumeBackgroundActivity</code></td>
<td>Resumes all content related activity.</td>
</tr>
<tr>
<td><code>RetryLimit</code></td>
<td>A read/write integer that contains the number of times to try and crawl an item.</td>
</tr>
<tr>
<td><code>SearchDatabaseCleanup</code></td>
<td>Cleans up the search database. You can enable or disable alerts by passing in a Boolean.</td>
</tr>
<tr>
<td><code>SetDefaultGatheringAccount</code></td>
<td>Sets the content crawler account to the user account and password you pass to the method.</td>
</tr>
</tbody>
</table>
Creating a New Content Source

In order to create new content sources, you have to understand that SharePoint breaks the content sources into types such as `SharePointContentSource`, `WebContentSource`, and even a `CustomContentSource`. Table 11-3 describes the content sources that you can create:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BusinessDataContentSource</td>
<td>BDC content source.</td>
</tr>
<tr>
<td>ContentSource</td>
<td>Base class for all content sources.</td>
</tr>
<tr>
<td>CustomContentSource</td>
<td>Used for your own custom content sources that you have a protocol handler built for.</td>
</tr>
<tr>
<td>ExchangePublicFolderContentSource</td>
<td>Exchange public folder content source.</td>
</tr>
<tr>
<td>FileShareContentSource</td>
<td>File share content source.</td>
</tr>
<tr>
<td>HierachicalContentSource</td>
<td>Base class for certain content sources, such as the Exchange public folder, File Share, and other hierarchical datasources.</td>
</tr>
<tr>
<td>LotusNotesContentSource</td>
<td>Lotus Notes content source.</td>
</tr>
<tr>
<td>SharePointContentSource</td>
<td>SharePoint, both WSS and Microsoft Office SharePoint Server, content source.</td>
</tr>
<tr>
<td>WebContentSource</td>
<td>Web site content source. Could also be used for SharePoint sites, but you should use the dedicated SharePoint content source because of the incremental crawl built-in to that content source.</td>
</tr>
</tbody>
</table>

The main process for creating the content source is consistent across all the content sources in that you call the `Create` method on your `ContentSourceCollection` and pass in a string value that specifies the type of content source. Where the differences come in are in the properties that are supported by the different content sources for the different datasources. Also, you must call the `Update` method after modifying the properties of your content source to send the changes to SharePoint. Let’s step through some of the content sources and the different properties they support.

Custom, File Share, Exchange, and Lotus Notes

These content sources support the ability to crawl subdirectories. You can enable or disable this capability by setting the `FollowDirectories` property.

Business Data Catalog Content Source

The Business Data Catalog (BDC) does not have special properties for you to set for it, but it does have some special methods to help you figure out the start address, since the BDC uses a Globally Unique Identifier (GUID). You call the `ConstructStartAddress` method and pass the name of your search application and the name of your BDC application. This method will return the start address for your BDC application, so that you can pass it as part of your content source creation. There is also a method
Chapter 11: Building Applications with the Search API and Web Services

called ParseStartAddress, which parses the BDC start address to get back the search application name and BDC application name.

SharePoint Content Source

With the SharePoint content source, you can specify whether you want to crawl just the sites under the start address for the content source or every site in the web application specified. The SharePointCrawlBehavior property controls this behavior.

Web Content Source

For the web content source, you can control the page and site enumeration depth. For example, you may only want to support three site hops and a depth of four pages. The MaxSiteEnumerationDepth property controls the site hops, and the MaxPageEnumerationDepth property controls the page depth.

Sample Code: Creating a Content Source

The sample code that follows creates three content sources to show you the process for creating a content source. It also shows the different properties you can set on the content sources. The code creates a file share, SharePoint, and web content source.

```csharp
using (SPSite site = new SPSite("http://moss"))
{
    SearchContext sc = SearchContext.GetContext(site);
    //Get the Content object for the SSP search context
    Content content = new Content(sc);
    //Retrieve the Content Source Collection
    ContentSourceCollection cscollection = content.ContentSources;
    //Create a File Share Content Source
    //Start address is \server\fileshare
    FileShareContentSource fscs = (FileShareContentSource)cscollection.Create(typeof(FileShareContentSource), "MOSS File Share");
    fscs.StartAddresses.Add(new Uri("\\moss\\demo");
    fscs.FollowDirectories = true;
    fscs.Update();
    //Create a SharePoint Content Source
    SharePointContentSource spcs = (SharePointContentSource)cscollection.Create(typeof(SharePointContentSource), "MOSS Site");
    spcs.StartAddresses.Add(new Uri("http://moss/"));
    spcs.SharePointCrawlBehavior = SharePointCrawlBehavior.CrawlSites;
    spcs.Update();
    //Create a web content source
    WebContentSource webcs = (WebContentSource)cscollection.Create(typeof(WebContentSource), "Microsoft.com");
    webcs.StartAddresses.Add(new Uri("http://www.microsoft.com"));
```
Chapter 11: Building Applications with the Search API and Web Services

webcs.MaxPageEnumerationDepth = 4;
webcs.MaxSiteEnumerationDepth = 2;
webcs.Update();
}

Enumerating Existing Content Sources

To enumerate your existing content sources, you can use a loop to scroll through the content sources or, if you want to find a particular content source, you can retrieve the content source by its ID or its name directly from the collection. The following code enumerates all the content sources and also shows how to retrieve a specific content source and get properties from it:

SearchContext sc = SearchContext.GetContext(site);
//Get the Content object for the SSP search context
Content content = new Content(sc);
//Retrieve the Content Source Collection
ContentSourceCollection cscollection = content.ContentSources;

//Enumerate all content types
foreach (ContentSource cs in cscollection)
{
    MessageBox.Show(cs.Name + ": " + cs.GetType());
}

//Retrieve a single content source
ContentSource cs2 = cscollection["Microsoft.com"];  
//Grab all the start addresses
StringBuilder startaddresses = new StringBuilder();
foreach(Uri startaddress in cs2.StartAddresses)
{
    startaddresses.AppendLine(startaddress.ToString());
}
MessageBox.Show(cs2.Name + ": " + startaddresses);

Deleting a Content Source

Deleting a content source is as easy as calling the delete method on your content source. The code that follows shows how to delete a content source:

//Delete the content source
cs2.Delete();

Crawling Content Sources

Besides creating and deleting content sources, you may actually want to crawl content in your content source! The content source object provides a number of methods to work with crawls, and a number of properties used to understand the status of your crawls. You can also programmatically control the crawl schedule for your content source. Table 11-4 outlines the properties and methods supported for crawling your content sources:
### Table 11-4

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CrawlCompleted</td>
<td>A datetime property that returns the last time the crawl completed.</td>
</tr>
<tr>
<td>CrawlStarted</td>
<td>A datetime property that returns the time the crawl started.</td>
</tr>
<tr>
<td>CrawlStatus</td>
<td>A crawlstatus enumeration that returns the current status such as CrawlingFull, CrawlingIncremental, Idle, Paused, and so on.</td>
</tr>
<tr>
<td>CrawlThrottle</td>
<td>Internal use only. Do not use.</td>
</tr>
<tr>
<td>FullCrawlSchedule</td>
<td>Returns a schedule class, so you can get or set the full crawl schedule.</td>
</tr>
<tr>
<td>IncrementalCrawlSchedule</td>
<td>Returns a schedule class, so you can get or set the incremental crawl schedule.</td>
</tr>
<tr>
<td>PauseCrawl</td>
<td>Pauses the active crawl.</td>
</tr>
<tr>
<td>ResumeCrawl</td>
<td>Resumes the active crawl that is paused.</td>
</tr>
<tr>
<td>StartFullCrawl</td>
<td>Starts a full crawl immediately.</td>
</tr>
<tr>
<td>StartIncrementalCrawl</td>
<td>Starts an incremental crawl immediately.</td>
</tr>
<tr>
<td>StopCrawl</td>
<td>Stops a crawl that is in progress.</td>
</tr>
</tbody>
</table>

To work with schedules, you will use classes, such as DailySchedule, WeeklySchedule, MonthlySchedule, or MonthlyDayOfWeekSchedule. You will create one of these objects and then pass that to the IncrementalCrawlSchedule or FullCrawlSchedule property to set the crawl schedule. The code that follows shows you how to crawl a content source and set up a crawl schedule. You can remove the crawl schedule by setting the incremental or full crawl schedule properties to null.

```csharp
SearchContext sc = SearchContext.GetContext(site);

//Get the Content object for the SSP search context
Content content = new Content(sc);

//Retrieve the Content Source Collection
ContentSourceCollection cscollection = content.ContentSources;

//Retrieve a Content Source
ContentSource cs = cscollection["Local Office SharePoint Server sites"]; cs.StartIncrementalCrawl();
MessageBox.Show(cs.CrawlStatus.ToString());

//Create a weekly incremental crawl schedule
WeeklySchedule wsschedule = new WeeklySchedule(sc); wsschedule.BeginDay = 1;
```

296
Chapter 11: Building Applications with the Search API and Web Services

//Start in January
wsschedule.BeginMonth = 1;

//Start in the Current Year
wsschedule.BeginYear = DateTime.Now.Year;

//Start at 11pm
wsschedule.StartHour = 23;
wsschedule.StartMinute = 00;

//Crawl every week
wsschedule.WeeksInterval = 1;

//Set the content source
cs.IncrementalCrawlSchedule = wsschedule;

//To clear the crawl schedule, set it to null
//cs.IncrementalCrawlSchedule = null;
cs.Update();

Working with Federated Locations

SharePoint provides the LocationConfigurationCollection object on the SearchContext object to allow you to add, query, import, export, and delete federated locations. The LocationConfigurationCollection contains LocationConfiguration objects that expose properties and methods for you to work with. Table 11-5 outlines the properties and methods on the LocationConfigurationCollection.

<table>
<thead>
<tr>
<th>Table 11-5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>Add</td>
</tr>
<tr>
<td>Contains</td>
</tr>
<tr>
<td>Remove</td>
</tr>
</tbody>
</table>

(Continued)
Chapter 11: Building Applications with the Search API and Web Services

**Table 11-5 (Continued)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TryGetLocationConfigurationByInternalName</td>
<td>Returns a LocationConfiguration object that matches the internal name of the location with the string you pass.</td>
</tr>
<tr>
<td>TryGetLocationConfigurationByName</td>
<td>Returns a LocationConfiguration object that matches the display name of the location with the string you pass.</td>
</tr>
<tr>
<td>CrawlProxy</td>
<td>A System.Net.WebProxy object that is the crawl proxy for your locations.</td>
</tr>
<tr>
<td>ProxyURL</td>
<td>Returns your proxy URL.</td>
</tr>
<tr>
<td>UseCrawlProxy</td>
<td>Boolean that specifies whether to use the crawl proxy.</td>
</tr>
</tbody>
</table>

You can also work directly with the federated locations in the collection which are LocationConfiguration objects. On these objects, you can import and export XML definitions of federated locations or change the properties, such as the name, URL, authentication information or other properties of your federated locations. Table 11-6 lists the properties and methods for the LocationConfiguration object:

**Table 11-6**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export</td>
<td>Exports the location definition to the System.IO.Stream that you pass to the method.</td>
</tr>
<tr>
<td>ExportToXML</td>
<td>Returns a System.Xml.XmlDocument that contains the definition of your federated location.</td>
</tr>
<tr>
<td>Import</td>
<td>Imports and creates a LocationConfiguration object from the System.IO.Stream object that you pass. You need to add your LocationConfiguration object to the LocationConfigurationCollection, in order to add it as a saved federated location.</td>
</tr>
<tr>
<td>Update</td>
<td>Call this method after you update your properties on your object.</td>
</tr>
<tr>
<td>AdminDescription</td>
<td>String that specifies the administrative description.</td>
</tr>
<tr>
<td>AllowedSiteCollectionGuids</td>
<td>A collection of GUIDs for the site collections that the location can be used on.</td>
</tr>
<tr>
<td>AllowedSiteCollectionUrls</td>
<td>A collection of string URLs that the location can be used on.</td>
</tr>
</tbody>
</table>

(Continued)
### Chapter 11: Building Applications with the Search API and Web Services

#### Table 11-6 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthInfo</td>
<td>Allows you to set authentication information using the AuthenticationInformation class.</td>
</tr>
<tr>
<td>Author</td>
<td>Name of the author who created the location.</td>
</tr>
<tr>
<td>ConnectionUrlTemplate</td>
<td>The URL of federated location to query for the results.</td>
</tr>
<tr>
<td>FullVisualization</td>
<td>The Visualization object for the location. This is the XSLT to use to transform results.</td>
</tr>
<tr>
<td>InternalName</td>
<td>The internal name for the location.</td>
</tr>
<tr>
<td>IsDeletable</td>
<td>A returned Boolean that specifies whether the location is deletable.</td>
</tr>
<tr>
<td>IsPrefixPattern</td>
<td>Boolean that specifies whether to use a prefix pattern for the location trigger.</td>
</tr>
<tr>
<td>IsRestrictedLocation</td>
<td>Boolean that specifies whether this location is restricted for use only on certain sites or site collections.</td>
</tr>
<tr>
<td>Languages</td>
<td>List of languages that are supported.</td>
</tr>
<tr>
<td>MoreLinkTemplate</td>
<td>URL to use for the More Results link.</td>
</tr>
<tr>
<td>Name</td>
<td>Display name of the federated location.</td>
</tr>
<tr>
<td>ProxyUrl</td>
<td>URL to the proxy to use, if any, for the location.</td>
</tr>
<tr>
<td>QueryReformatPattern</td>
<td>The pattern that the query must match, in order for the federated search to be triggered.</td>
</tr>
<tr>
<td>QueryRestriction</td>
<td>A query restriction such as a fixed query.</td>
</tr>
<tr>
<td>SummaryVisualization</td>
<td>The Visualization object that contains the XSLT for formatting the summary results from your federated query.</td>
</tr>
<tr>
<td>TopAnswerVisualization</td>
<td>The Visualization object that contains the XSLT for formatting the top results from your federated query.</td>
</tr>
<tr>
<td>Type</td>
<td>Location type using the LocationType enumeration, which can be Custom, LocalSharePoint, OpenSearch, SharePointWebService, or XmlSearch.</td>
</tr>
<tr>
<td>Version</td>
<td>A System.Version that contains the version number for your configuration.</td>
</tr>
<tr>
<td>Visualizations</td>
<td>A collection of Visualization objects that contains all the visualizations/XSLTs for your search results.</td>
</tr>
</tbody>
</table>
Chapter 11: Building Applications with the Search API and Web Services

The following code shows you how to enumerate your locations and export them out to XML:

```csharp
using (SPSite spsite = new SPSite("http://moss"))
{
    SearchContext sc = SearchContext.GetContext(spsite);
    LocationConfigurationCollection locations = sc.LocationConfigurations;
    foreach (LocationConfiguration location in locations)
    {
        MessageBox.Show("Name: " + location.Name + " URL: " +
                        location.ConnectionUrlTemplate);
        //Export to XML and display
        MessageBox.Show(location.ExportToXml().OuterXml.ToString());
    }
}
```

Crawl Rules and File Types

When working with content, sometimes you want to create rules that include or exclude certain content, specify custom authentication, or provide custom security trimming. The same applies to file types, in that you may want to include or exclude certain types. SharePoint provides an object model for working with crawl rules through the `CrawlRuleCollection` and `CrawlRule` classes, and for file types through the `ExtensionCollection` and `Extension` classes.

Creating a Crawl Rule

To create a crawl rule, first you need to get the `CrawlRule` collection from your `Content` object. Then, you can use the `Create` method to create your crawl rule. The `Create` method takes two parameters. The first is an enumeration of the type of crawl rule that can be either an `ExclusionRule` or an `InclusionRule` to exclude or include content, respectively. The second is a string for the path of the content that you want to have the rule work with. Once you create your new rule, you can modify the properties of the returned `CrawlRule` object to make modifications to your settings. Table 11-7 outlines the important properties and methods for the `CrawlRule` class:

Table 11-7

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountLastModified</td>
<td>A datetime property that returns the last time the account information was modified for the rule.</td>
</tr>
<tr>
<td>AccountName</td>
<td>A string specifying the name of the account used for the rule. Use <code>SetCredentials</code> to set the credentials used.</td>
</tr>
<tr>
<td>Activate</td>
<td>Reactivates the crawl rule after you disable it.</td>
</tr>
<tr>
<td>AuthenticationData</td>
<td>Returns the authentication data for your crawl rule.</td>
</tr>
</tbody>
</table>

(Continued)
### Table 11-7 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticationPath</td>
<td>Returns the authentication path for your rule.</td>
</tr>
<tr>
<td>AuthenticationType</td>
<td>Enumeration of the type of authentication. This can be different types, such as basic auth, certificate, cookie, default, form, or NTLM.</td>
</tr>
<tr>
<td>ContentClass</td>
<td>Gets or sets the content class sent to the protocol handler for matches. The out-of-the-box protocol handlers do not use this.</td>
</tr>
<tr>
<td>CrawlAsHttp</td>
<td>Specifies whether to crawl the content as HTTP. This is used with hierarchical content sources.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the crawl rule.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Boolean that specifies whether the rule is enabled or disabled.</td>
</tr>
<tr>
<td>ErrorPages</td>
<td>Returns a string that is the error pages to access if authentication fails for authentication types, such as cookie or forms authentication.</td>
</tr>
<tr>
<td>FollowComplexUrls</td>
<td>Boolean that specifies whether the crawler should follow complex URLs, such as URLs having a ? in their path.</td>
</tr>
<tr>
<td>Method</td>
<td>Returns the method to use to retrieve authentication data for Forms based authentication.</td>
</tr>
<tr>
<td>Path</td>
<td>Gets or sets the path that the rule applies to.</td>
</tr>
<tr>
<td>PluggableSecurityTrimmerId</td>
<td>Gets or sets the ID of the security trimmer the rule applies to.</td>
</tr>
<tr>
<td>Priority</td>
<td>Gets or sets the priority of the rule. Remember that SharePoint will use the first rule that matches, so getting the priority order correct is important.</td>
</tr>
<tr>
<td>SetCredentials</td>
<td>Sets the credentials used to access items. You can have the default account, certificate, cookie, form, NTLM, or basic authentication types.</td>
</tr>
<tr>
<td>SuppressIndexing</td>
<td>Boolean that specifies whether to add the content to the index.</td>
</tr>
<tr>
<td>Test</td>
<td>Returns a Boolean that tests whether the passed URL triggers the crawl rule.</td>
</tr>
<tr>
<td>Type</td>
<td>Gets or sets the type of crawl rule whether it is an inclusion or an exclusion rule.</td>
</tr>
<tr>
<td>Update</td>
<td>Call this method after making any changes to your properties.</td>
</tr>
</tbody>
</table>
Chapter 11: Building Applications with the Search API and Web Services

The following code creates a crawl rule that specifies a different NTLM account to crawl the content:

```csharp
CrawlRuleCollection crc = content.CrawlRules;
CrawlRule cr = crc.Create(CrawlRuleType.InclusionRule, "http://intranet");
string spwd = "pass@word1";
char[] pwd = spwd.ToCharArray();
foreach (char c in pwd)
{
    securepassword.AppendChar(c);
}
cr.SetCredentials(CrawlRuleAuthenticationType.NTLMAccountRuleAccess, "LITWAREINC\patc", securepassword);
```

**Working with File Types**

File types tell the crawler which files to include or exclude from the index. To work with file types, you will work with the `ExtensionCollection` and the `Extension` class. You can create new file types by using the `Create` method on the `ExtensionCollection` object and pass in a string for your extension. You will need an `IFilter` for the extension that you create. The following code creates a new extension and enumerates all the extensions:

```csharp
//Get the extension collection
ExtensionCollection ec = content.ExtensionList;

Extension customextension = ec.Create("dud");

foreach (Extension ex in ec)
{
    MessageBox.Show(ex.FileExtension);
}
```

**Scopes**

Just as you can work with scopes through the user interface, the object model provides a rich way to work with scopes. To work with scopes, you will use the `Scopes` class and all of its children. You create your `Scopes` object by instantiating a new object from the `Scopes` class and passing in your `SearchContext` object. Table 11-8 lists the properties and methods of the `Scopes` class:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllDisplayGroups</td>
<td>Returns a <code>ScopeDisplayGroupCollection</code> object.</td>
</tr>
<tr>
<td>AllScopes</td>
<td>Returns a <code>ScopeCollection</code> object corresponding to all the scopes.</td>
</tr>
<tr>
<td>AverageCompilationDuration</td>
<td>Returns a <code>TimeSpan</code> object that represents the average time to compile your scopes.</td>
</tr>
<tr>
<td>CompilationPercentComplete</td>
<td>Returns an integer that represents the percent complete of the compilation process for your scopes.</td>
</tr>
</tbody>
</table>

(Continued)
Table 11-8 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompilationScheduleType</td>
<td>Gets or sets the schedule type, which can be automatic, custom or none.</td>
</tr>
<tr>
<td>CompilationState</td>
<td>Returns the compilation state, which can be compiling or idle.</td>
</tr>
<tr>
<td>GetDisplayGroup</td>
<td>Returns a ScopeDisplayGroup for the provided URL and display group name.</td>
</tr>
<tr>
<td>GetDisplayGroupsForSite</td>
<td>Returns an enumerator to loop through all the ScopeDisplayGroup instances for the URL of the site you provide.</td>
</tr>
<tr>
<td>GetScope</td>
<td>Returns a Scope class instance for the site URL you specify and a name.</td>
</tr>
<tr>
<td>GetScopesForSite</td>
<td>Returns an enumerator of the Scopes instances for the site URL you provide.</td>
</tr>
<tr>
<td>GetSharedScope</td>
<td>Returns the Scope instance corresponding to the name you specify.</td>
</tr>
<tr>
<td>GetSharedScopes</td>
<td>Returns an enumerator to loop through all the shared scopes.</td>
</tr>
<tr>
<td>GetUnusedScopesForSite</td>
<td>Returns an enumerator, so you can loop through all the site scopes that are not part of a display group for a site URL that you provide.</td>
</tr>
<tr>
<td>LastCompilationTime</td>
<td>Returns a datetime that is the last time compilation occurred.</td>
</tr>
<tr>
<td>NextCompilationTime</td>
<td>Returns a datetime that is the next time compilation will happen.</td>
</tr>
<tr>
<td>ScopesNeedingCompilation</td>
<td>Returns an integer of the number of scopes that need compiling.</td>
</tr>
<tr>
<td>StartCompilation</td>
<td>Starts the compilation process for scopes.</td>
</tr>
<tr>
<td>Update</td>
<td>Call the method after updating any properties on the instance.</td>
</tr>
</tbody>
</table>

Creating a scope is just a matter of calling the Create method on the ScopeCollection class and passing in values, such as the name of the scope, whether it is a site scope or a global scope, alternate results pages, and a few other properties.

Once you create your scope, you need to add rules to it, so that SharePoint can know which type of items to add to the scope. This is accomplished using the ScopeRuleCollection and ScopeRule object. You can specify an all content rule, property rules, or URL-based rules. All content is self-explanatory. Property rules allow you to specify managed properties, and corresponding values to use to filter, based on those properties. URL rules allow you to specify a URL as your filter.
Chapter 11: Building Applications with the Search API and Web Services

You will notice in the following sample code that the `ScopeRuleCollection` instance is retrieved off the `Scope` object. From there, you can add, modify, delete or enumerate your rules.

```csharp
//Create a new scope
ScopeCollection scopeColl = scopes.AllScopes;
Scope scope = scopeColl.Create("Author is Jesse", "All documents authored by Jesse Meriam", null, true, "results.aspx", ScopeCompilationType.AlwaysCompile);

//Create some rules
ScopeRuleCollection scoperules = scope.Rules;

//Get the managed property
Schema sspSchema = new Schema(sc);
ManagedPropertyCollection properties = sspSchema.AllManagedProperties;
ManagedProperty author = properties["Author"]; 
scoperules.CreatePropertyQueryRule(ScopeRuleFilterBehavior.Include, author, "Jesse Meriam");
scopes.StartCompilation();

//Enumerate all scopes and rules
foreach (Scope scope in scopes.AllScopes)
{
    ScopeRuleCollection rules = scope.Rules;
    foreach (ScopeRule rule in rules)
    {
        //Figure out the type
        if (rule.RuleType == ScopeRuleType.PropertyQuery)
        {
            PropertyQueryScopeRule prule = (PropertyQueryScopeRule)rule;
            MessageBox.Show("Property: " + prule.Property.Name + " Value: " + prule.Value);
        }
        MessageBox.Show(rule.FilterBehavior + " " + rule.RuleType);
    }
}
```

**Schema Management**

You can work with the schema of Search programmatically through the object model. This includes finding the crawled properties that the search indexer has found and mapping those crawled properties to managed properties for use in your searches. You work with the schema through the Schema object, which you create an instance of in your code, and pass the search context as a parameter.
Chapter 11: Building Applications with the Search API and Web Services

Once you have the Schema object, you will see a number of properties and methods on the object. Table 11-9 describes these properties and methods:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllCategories</td>
<td>Returns a CategoryCollection object for all the categories for a SSP.</td>
</tr>
<tr>
<td>AllManagedProperties</td>
<td>Returns a ManagedPropertyCollection object for all the managed properties for a SSP.</td>
</tr>
<tr>
<td>GetCrawledProperty</td>
<td>Returns a crawled property based on the propset, name, and type.</td>
</tr>
<tr>
<td>QueryCrawledProperties</td>
<td>Allows you to query all crawled properties in a filtered way, since the collection could be big. You can pass a string filter, or if you pass an empty string, you will get all properties in alphabetical order. Also, you can pass the maximum number of properties to return. An example of parameters is (string.Empty, 500, Guid.NewGuid(), string.Empty, true), which would return the first 500 properties.</td>
</tr>
</tbody>
</table>

**Querying All Categories**

You can query all your categories by retrieving the CategoryCollection object from the Schema object’s AllCategories property. Once you have the collection, you can enumerate the collection to find all the categories and all the properties in that category. You can also specify whether the category should autocreate new managed properties from crawl properties, turn off crawled property discovery for the category, or enumerate unmapped crawled properties, so that you can map them to managed properties. The code that follows enumerates all the categories and returns the name and crawled property count for each:

```csharp
foreach (Category category in cc) {
    MessageBox.Show(category.Name + " * * + category.CrawledPropertyCount);
}
```

**Querying All Crawled Properties**

You can query all your crawled properties by using the QueryCrawledProperties method and passing the right parameters to the query. This method takes the filter, number of properties to return, GUID, last property name, and a Boolean, which is the direction. The following sample returns the first 1000 properties in the collection. From there, you can use the CrawledProperty methods and properties to work with your crawled properties.

```csharp
foreach (CrawledProperty cp in schema.QueryCrawledProperties(string.Empty, 1000, Guid.NewGuid(), string.Empty, true)) {
    MessageBox.Show(cp.Name + " * + cp.VariantType + " + cp.IsMappedToContents);
}
```
Chapter 11: Building Applications with the Search API and Web Services

Querying All Managed Properties

To retrieve all your managed properties, you just need to use the AllManagedProperties property on your schema object. From there, you can work with the properties and methods of the ManagedProperty object. The following code enumerates all the managed properties in an SSP.

```csharp
ManagedPropertyCollection mpc = schema.AllManagedProperties;
foreach (ManagedProperty mp in mpc)
{
    MessageBox.Show(mp.Name + " " + mp.Description + " " + mp.PID);
}
```

Creating and Mapping a New Managed Property

To create a new managed property is a two step process. First, you first need to create your new managed property. Second, you need to map a crawled property to the managed property you created. To add a new managed property, use the Create method on the ManagedPropertyCollection object. This method takes the name and type of the managed property. There is an enumeration that will help you with the type to specify.

Once you have your managed property, you can set other properties, such as whether the managed property is retrievable as a search result, whether it can be used in scopes, and whether it can be queried. Remember to call the Update method if you change any of the properties to persist your changes.

Next, you need to create a MappingCollection object. You will add Mapping objects to your MappingCollection object that will be the mappings between your crawled and managed properties. You need to retrieve the crawled properties that you want to map, and we recommend that you use the QueryCrawledProperties method with a string filter to do that. You will see an example of this in the code. Once you get your crawled properties, you will want to get the property set, name, and variant type, since you need to pass those to the Mapping object that you will create.

To create the Mapping objects that you will add to your MappingCollection object, just create a new Mapping object directly, since it has a constructor that takes the crawled property set, crawled property name, the crawled variant type, and the property set id (PID) of the managed property. Your newly instantiated object will have the mapping established between the crawled and managed property.

The next step is to add your Mapping object to your MappingCollection. Use the Add method of the MappingCollection to add your MappingObject.

Finally, call the SetMappings method on your ManagedProperty object to make the mapping persist on your managed property. The code that follows shows all of these steps:

```csharp
ManagedPropertyCollection mpc = schema.AllManagedProperties;
ManagedProperty mp = mpc.Create("Geography", ManagedDataType.Text);
//Set some properties to make it retrievable, scopeable and queryable
mp.Retrievable = true;
mp.EnabledForScoping = true;
mp.FullTextQueriable = true;
```
Chapter 11: Building Applications with the Search API and Web Services

//We call update later
//Make the mapping between a crawled property and this property
MappingCollection mapcoll = new MappingCollection();

Guid cppropset = new Guid();
string propname = "";
int varianttype = 0;

//Query our crawled properties to retrieve the right crawled property
//I'm using Item URL here just as an example
foreach (CrawledProperty cp in schema.QueryCrawledProperties("ItemURL", 1000, Guid.NewGuid(), string.Empty, true))
{
    //Retrieve what we need to create a property mapping
    cppropset = cp.Propset;
    propname = cp.Name;
    varianttype = cp.VariantType;
}

//Create a mapping object to use to add to the collection
Mapping map = new Mapping(cppropset, propname, varianttype, mp.PID);

//Add it to the MappingCollection
mapcoll.Add(map);

//Set the mapping
mp.SetMappings(mapcoll);
mp.Update();

Changing Property Weights

Property weightings are one area where the functionality is only available through the object model and
not the user interface. Property weightings allow you to customize how the different properties are
weighted in the relevancy calculations. One warning is that if you get the weighting wrong, you could
make your results very irrelevant, which is the opposite of what you are probably trying to achieve. You
should tread very carefully here and make sure that you test any changes before applying them into
production. The code below shows you how to retrieve the rankings for your managed properties:

Schema schema = new Schema(sc);
ManagedPropertyCollection mpc = schema.AllManagedProperties;

foreach (ManagedProperty mp in mpc)
{
    MessageBox.Show(mp.Name + " * " + mp.Weight.ToString());
}

//If you want to update, just change the weight which is a float and call Update()
Chapter 11: Building Applications with the Search API and Web Services

Editorial Results and Ranking

Many times, when users search for something, they need help finding what they are looking for by returning to highly relevant results, either that other humans have put into the system, or that are synonyms of the term they are looking for. Programmatically, the SharePoint object model provides a way to work with keywords, best bets, and synonyms through the API. For keywords, you need to create a Keywords object and pass in your search context and the site collection for which you want to retrieve keywords. For best bets, you use the BestBets property. For synonyms, you use the Synonyms property. Table 11-10 shows the properties and methods for the Keywords object:

Table 11-10

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllKeywords</td>
<td>Returns a KeywordCollection object for all the keywords for the site. You can retrieve best bets and synonyms from the Keyword objects in this collection.</td>
</tr>
<tr>
<td>GetAllBestBets</td>
<td>Returns a BestBetCollection, which contains all the best bets for the site.</td>
</tr>
<tr>
<td>GetFilteredBestBets</td>
<td>Returns a BestBetCollection based on your filter, which could be filtered by title, URL, both, or none.</td>
</tr>
<tr>
<td>GetFilteredKeywords</td>
<td>Returns a KeywordCollection based on your filter, which can specify a view such as all keywords, expired keywords, or keywords needing review. You can also pass a KeyworldFilter enumeration, which can be by best bet title, URL, contact, keyword, none, or synonym. Finally, you pass the string to match to the filter.</td>
</tr>
</tbody>
</table>

Once you get a KeywordCollection, you can create new keywords from the collection by using the Create method, or you can delete, change, or enumerate your keywords. Once you create or retrieve a keyword using the Keyword class, you can start working with the properties and methods of this class. Table 11-11 outlines the properties and methods for the Keyword class:

Table 11-11

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BestBets</td>
<td>Returns the BestBetCollection object for the keyword.</td>
</tr>
<tr>
<td>Contact</td>
<td>Gets or sets the contact for the keyword.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the keyword.</td>
</tr>
<tr>
<td>EndDate</td>
<td>Get or set the end date for the keyword to expire using a datetime type.</td>
</tr>
<tr>
<td>ReviewDate</td>
<td>Get or set the review time for the keyword using a datetime type.</td>
</tr>
<tr>
<td>StartDate</td>
<td>Get or set the date and time a keyword should go live.</td>
</tr>
<tr>
<td>Synonyms</td>
<td>Gets the SynonymCollection for all the synonyms for the keyword.</td>
</tr>
<tr>
<td>Term</td>
<td>Get or set the term the keyword describes.</td>
</tr>
<tr>
<td>Update</td>
<td>Call this method after making any changes on the properties for your keyword.</td>
</tr>
</tbody>
</table>
Best bets are also pretty straightforward once you retrieve them from your keyword. You can add, delete, or modify your best bets. The key properties on a BestBet object are the Description, which gets or sets the description for the best bet; the ParentKeywords property, which returns a KeywordCollection of all the keywords associated with the best bet; the Title property, which gets or sets the title for the best bet; and the Url property, which gets or sets the URL for the best bet. Remember to call the Update method after working with any of the properties on your best bet.

For synonyms, it is even simpler. The only property you need to set for your synonym is the Term property, which is the synonym term to use, such as MSFT and Microsoft.

The following code creates a keyword, best bet, and synonym, and enumerates all of these collections to show that the creation was successful:

```csharp
//Create an instance of the Keywords class
Keywords keywords = new Keywords(sc, new Uri("http://moss"));

//Retrieve all keywords for the site using AllKeywords
KeywordCollection kccoll = keywords.AllKeywords;

//Add a new keyword
Keyword keyword = kccoll.Create("Microsoft", DateTime.Now);

//Change some of the properties
keyword.Definition = "Microsoft Corporation Nasdaq MSFT";
keyword.Contact = "LITWAREINC\patc";
keyword.Update();

//Add a new Best Bet
BestBetCollection bb = keyword.BestBets;

//Add a new Synonym
SynonymCollection scoll = keyword.Synonyms;
scoll.Create("Microsoft SharePoint");

StringBuilder _results = new StringBuilder();
foreach(Keyword kw in kccoll)
{
    _results.AppendLine("Keyword: " + kw.Term + "=" + kw.Definition);
    foreach(BestBet bestbet in kw.BestBets)
    {
        _results.AppendLine("Best Bet: " + bestbet.Title);
        _results.AppendLine("URL: " + bestbet.Url);
    }
    _results.AppendLine();
}
```

Chapter 11: Building Applications with the Search API and Web Services

Best bets are also pretty straightforward once you retrieve them from your keyword. You can add, delete, or modify your best bets. The key properties on a BestBet object are the Description, which gets or sets the description for the best bet; the ParentKeywords property, which returns a KeywordCollection of all the keywords associated with the best bet; the Title property, which gets or sets the title for the best bet; and the Url property, which gets or sets the URL for the best bet. Remember to call the Update method after working with any of the properties on your best bet.

For synonyms, it is even simpler. The only property you need to set for your synonym is the Term property, which is the synonym term to use, such as MSFT and Microsoft.

The following code creates a keyword, best bet, and synonym, and enumerates all of these collections to show that the creation was successful:

```csharp
//Create an instance of the Keywords class
Keywords keywords = new Keywords(sc, new Uri("http://moss"));

//Retrieve all keywords for the site using AllKeywords
KeywordCollection kccoll = keywords.AllKeywords;

//Add a new keyword
Keyword keyword = kccoll.Create("Microsoft", DateTime.Now);

//Change some of the properties
keyword.Definition = "Microsoft Corporation Nasdaq MSFT";
keyword.Contact = "LITWAREINC\patc";
keyword.Update();

//Add a new Best Bet
BestBetCollection bb = keyword.BestBets;

//Add a new Synonym
SynonymCollection scoll = keyword.Synonyms;
scoll.Create("Microsoft SharePoint");

StringBuilder _results = new StringBuilder();
foreach(Keyword kw in kccoll)
{
    _results.AppendLine("Keyword: " + kw.Term + "=" + kw.Definition);
    foreach(BestBet bestbet in kw.BestBets)
    {
        _results.AppendLine("Best Bet: " + bestbet.Title);
        _results.AppendLine("URL: " + bestbet.Url);
    }
    _results.AppendLine();
}
```
Chapter 11: Building Applications with the Search API and Web Services

```csharp
    {
        _results.AppendLine(bestbet.Title + " " + bestbet.Description + " " + bestbet.Url);
    }

    foreach (Synonym synonym in kw.Synonyms)
    {
        _results.AppendLine("Synonym: " + synonym.Term);
    }

    MessageBox.Show(_results.ToString());
    _results.Length = 0;
}
```

**Working with Authoritative Sites and Rankings**

The other part of editorializing your search and making it more relevant involves ranking. You can think of ranking as the score that the search system gives to content based on the search terms, the metadata, authoritative sites, and so on. You can influence the ranking by adding authoritative sites, demoting sites, or changing rank weightings for particular relevancy algorithms.

To work with rankings, you need to instantiate a new `Ranking` object in your code. The `Ranking` object takes your search context as a parameter. From there, you can use the `Ranking` object to promote, demote, or change relevancy.

To add a new authoritative site, you use the `AuthorityPages` property of the `Ranking` object with the `Create` method on that `AuthorityPages` collection. The `Create` method takes a URI of the site you want to promote, and a level, which is zero based. The lower the level, with zero being the lowest, the more important the site is from a relevancy standpoint. One thing to realize is if you create new authoritative pages, you need to make sure to call the `StartRankingUpdate` method; otherwise, your new authoritative page will not be used until the next ranking update is performed.

To add a demoted site is easy. You just call the `Create` method on your `DemotedSiteCollection` class and pass in the URI of the site you want to demote.

For the ranking parameters, you can enumerate and change the ranking parameters, but you cannot add new ones. Some of the ranking parameters include the weightings of the different file types, click distance weighting, and URL depth weighting. We won’t cover this in depth, since modifying these properties can have a very negative effect on your relevancy if you do not get it 100% right.

The following code shows you how to work with ranking and sites:

```csharp
    //Create an instance of the Ranking class
    Ranking ranking = new Ranking(sc);

    //Add a new authoritative page
    ranking.AuthorityPages.Create(new Uri("http://intranet"), 0);

    //Start the update
```
Chapter 11: Building Applications with the Search API and Web Services

ranking.StartRankingUpdate(RankingUpdateType.FullUpdate);

// Add a new demoted site

// Enumerate authoritative pages
foreach (AuthorityPage ap in ranking.AuthorityPages)
{
    MessageBox.Show("Authoritative: " + ap.Url + " " + ap.Level + " " + ap.Status);
}

foreach (DemotedSite ds in ranking.DemotedSites)
{
    MessageBox.Show("Demoted: " + ds.Url + " " + ds.CrawledDocumentCount);
}

RankParamCollection rpc = ranking.RankingParameters;

foreach (RankingParameter rp in rpc)
{
    MessageBox.Show("Ranking Param: " + rp.Name + " " + rp.Value);
}

Search Logs

We will not spend a lot of time on the API of the crawl log, since many of the applications you will build will probably be using the other parts of the administration API. The crawl log API allows you to get the crawl log and filter it by different characteristics, such as host URL, message type success or failure, date and time, and other filters. You can also purge the crawl log if you want to clean it up. You interact with the crawl log through the LogViewer object, which you instantiate in your code. Once you have the LogViewer object, you can work with its methods to peruse the crawl log. The following code shows how to get this object and use some of the methods to look at the crawl log:

// Create a new LogViewer object
LogViewer lv = new LogViewer(sc);

// Get all the status messages
System.Data.DataTable dt = lv.GetAllStatusMessages();

dataGridView1.DataSource = dt;
int nextstart = 0;


dataGridView1.DataSource = dt1;
Chapter 11: Building Applications with the Search API and Web Services

Propagation

SharePoint propagates the index from your search servers out to your query servers, so that you always have up-to-date information when users perform queries. You may want to check the propagation status programmatically to make sure that the propagation is working. SharePoint gives you a simple object model, as a way to view the all-up propagation status, and also as a way to enumerate all your query servers, so you can understand the status of those servers as well. To work with propagation, you need to instantiate the Propagation class and pass in your search context object. From there, you can retrieve the Status property, which returns a PropagationStatus enumeration, which tells you the status, such as idle, no propagation, propagating, waiting for query server, or waiting for initialization. You can also enumerate your query servers by getting the QueryServers property, which returns an IEnumerable, which you can use to scroll through the QueryServer objects in the collection. The following code shows you how to use the propagation API:

```csharp
Propagation prop = new Propagation(sc);

//Get the Status
PropagationStatus propstatus = prop.Status;
MessageBox.Show(propstatus.ToString());

//Enumerate the Query Servers
System.Collections.IEnumerable qs = prop.QueryServers;
foreach (QueryServer query in qs)
{
    MessageBox.Show(query.name + " " + query.propagationStatus + " " + query.readyState + " " + query.indexLocation);
}
```

Writing Search Applications

Now that you have seen how to write to the administration API, we will turn our attention to writing to the search object model so that you can perform searches against the index and get back your results. There are a number of different ways that you can programmatically access the search functionality. Which one you choose will depend on where your code is running, and what functionality you want to achieve. SharePoint provides the ability to customize a search by passing new parameters along the URL to the Search Center. You can write to the search object model for code running on the SharePoint Server, and you can use the search web services for code that runs off the SharePoint server. One thing we will cover, as well, that is not 100% search engine related, but is considered searching, is the Content Query web part, and how to use it to perform a search of your content.

Building Queries

The first things we’ll take a look at are the formats that you can use to build your queries against SharePoint, since the query and results are the main things with which you will interact. If you get the query wrong, nothing else matters with search. We will look at three key areas: keyword, URL, and SQL syntax.
Chapter 11: Building Applications with the Search API and Web Services

**Keyword Syntax for Searches**

The keyword syntax is the standard syntax that the search box uses in SharePoint. The keyword syntax is a new feature in SharePoint, since previous versions used SQL queries to perform searches. The move to a keyword syntax provides symmetry between SharePoint, Windows Desktop Search, and Windows Live. The keyword syntax consists of a property/value pair that you can customize.

**Inclusion/Exclusion**

For inclusion and exclusion, SharePoint supports the + and – characters, with + being inclusion and – being exclusion. When you have multiple search terms, inclusion is implied. The following example would search for Microsoft SharePoint but would leave off v2 and add 2007 and v3:

```
Microsoft SharePoint -v2 +2007 +v3
```

**Boolean Search**

SharePoint, by default, ANDs all your search terms together. Since this may not be the behavior that you want, you can explicitly tell it to OR certain search terms together, but this is only available through the search API or through the SQL syntax. If you attempt to put any of the logical operators in a keyword syntax query, such as OR, NEAR, or FAR, search will treat them as keywords.

**Property, Scope, and Collapsed Results/Duplicates Restrictions**

You can narrow your results by using restrictions. Restrictions are name/value pairs that allow you to specify filters for any managed property. One thing to note is that SharePoint only supports string and number properties for filtering. The restrictions support stemming, so that you can generate variations of the restricted property. Table 11-12 gives some examples of the different types of restrictions you can do:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Name</td>
<td>Allows you to specify a property/value to filter on, such as author:Tom, or author: &quot;Bill Gates&quot;.</td>
</tr>
<tr>
<td>Scope</td>
<td>Filter by scope such as scope:people or any of your custom scopes.</td>
</tr>
<tr>
<td>Duplicates</td>
<td>Filter by all the duplicates for an item. This will return the duplicates rather than the item itself. An example is duplicates:<a href="http://intranet/item.docx">http://intranet/item.docx</a>.</td>
</tr>
<tr>
<td>Item</td>
<td>Search for a specific item, such as item:<a href="http://intranet/item.docx">http://intranet/item.docx</a>.</td>
</tr>
<tr>
<td>File type</td>
<td>Filter by file type, such as filetype:pptx.</td>
</tr>
<tr>
<td>Site</td>
<td>Filter by site, such as site:<a href="http://intranet">http://intranet</a>.</td>
</tr>
<tr>
<td>Threads</td>
<td>Filter by threads for an item, such as thread: &lt;URL to the item&gt;.</td>
</tr>
<tr>
<td>Language variants</td>
<td>Filter by the language variants, such as langvar: &lt;URL to the item&gt;.</td>
</tr>
</tbody>
</table>
Chapter 11: Building Applications with the Search API and Web Services

A couple of things to note about the property filters: They do not support ranges or dates. They also do not support other comparison operators, such as greater than or less than. Finally, they do not support wildcard searches. The advanced search page supports partial word search through its property restrictions. If you need wildcard search, the SharePoint API supports it, just not the Out of the Box (OOB) web parts. You could design your own custom web part to perform the wildcard search, using the SQL syntax of the API.

SharePoint also supports multiple filters in a single keyword query that can either be filtering on the same property or on multiple properties. One thing to note is that if you filter on the same property, such as site:http://intranet site:http://extranet, SharePoint will OR the results together, so that you get a union of the results, not an intersection. If you have multiple filters on different properties, they will be the intersection of the results for those filters.

**URL Syntax**

SharePoint also supports using a URL-based syntax for you to be able to pass parameters along the querystring to the search application to perform specific searches. The best use for this capability is when you want to create a customized program to capture the URL and send that payload over to SharePoint. Imagine the scenario where you have a custom ASP.NET application that is not SharePoint based. You could collect information from the user and pass the parameters over to SharePoint to perform the search. You could even put the SharePoint search results into an IFRAME in your custom applications. The other use for URL syntax is for saving searches for future reference. For example, you may want to persist searches for the user, so that you can save their top 10 searches. You would do this by persisting the URL used for the search, and then replaying that URL, when they want to run the saved search.

The URL syntax is shown in Table 11-13. You will want to call the results.aspx ASP.NET page for your Search Center to perform your search.

**Table 11-13**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>Specifies the keywords. Make sure to URL encode your special characters when sending the URL, such as %20 for spaces.</td>
</tr>
<tr>
<td>s</td>
<td>Specifies the search scope. Separate multiple search scopes using %2c.</td>
</tr>
<tr>
<td>v</td>
<td>Specifies the view to use when returning results. There are two choices: either date or relevance for modified date, or relevance views.</td>
</tr>
<tr>
<td>start</td>
<td>Specifies which results page to show.</td>
</tr>
<tr>
<td>#</td>
<td>If you have multiple queries, you can customize the parameters to the query; for example, for query #2 on your page, you would append the parameter with the number 2.</td>
</tr>
</tbody>
</table>
Chapter 11: Building Applications with the Search API and Web Services

SQL Syntax
For complex searches, you can turn to the SQL syntax for search. Using the SQL syntax, you can use comparison operators, arbitrary groupings (using logical operators such as AND, OR, NOT), full-text search query terms (such as FREETEXT and CONTAINS), the LIKE term, the ORDER BY term, and wildcard searches. The SQL syntax is not as fully functional as SQL Server’s syntax and is more of a query than read/write syntax. For example, you cannot do INSERT, DELETE, or GRANT when using the search SQL syntax.

There are a couple of things you need to know before starting to work with the SQL syntax. SharePoint handles some of the datatypes differently when you are writing queries, from what you may be used to as part of the ANSI SQL standard.

String Literals
String literals must be enclosed in a pair of single quotation marks, and you can escape a single quotation, if you need one in your string, by putting in two single quotation marks together. For an empty string, put two single quotation marks together, like this: ‘ ’.

Numbers
Numbers should not be enclosed in single quotations. SharePoint supports positive, negative, decimal, currency, and scientific (4.5E-02) numbers.

Hexidecimal
SharePoint supports hex numbers. They should not be enclosed in single quotation marks.

Boolean
Booleans (such as true and false) should be written out and not included in quotation marks. SharePoint is case insensitive, so it does not matter if you use TRUE or true.

Dates
Dates should be enclosed in single quotation marks. You must put dates in the format year/month/day or year-month-day, and year must be a four digit value. Time must be specified in the format hours:minutes:seconds. You can also use relative times by using the DATEADD function, which we will cover later in the chapter.

SQL Search Query Syntax
The SQL search query syntax is pretty straightforward. The following snippet shows the basic structure of your SQL queries, which you will customize based on different operators in your queries:

```
SELECT <columns>
FROM <content source>
WHERE <conditions>
ORDER BY <columns>
```

Let’s step through each of the different statements in the syntax, so that you can see the different options that you have.
Chapter 11: Building Applications with the Search API and Web Services

**SELECT Statement**

The **SELECT** statement allows you to select the columns that you want returned from your query. You can have multiple columns that are different types. There are some differences between SharePoint and SQL Server syntax for the **SELECT** statement that you should be aware of. First, SharePoint does not support **SELECT ***, since there is no set of properties common to all the different contents that may be returned by your search. You may be wondering what properties you can query if you can’t say “get all properties” to the engine. You can discover properties using the API before performing your query, so that you know what’s available. Second, SharePoint does not support column aliasing. Finally, SharePoint does not support casting values from one datatype to another. A sample query is:

```
SELECT rank, title, author FROM scope() WHERE CONTAINS(author, 'Tom Rizzo')
```

**FROM Clause**

The **FROM** clause can only specify the default **Scope()** keyword. To specify a search scope, you now specify the scope in your **WHERE** clause using the syntax **"scope" = <scopename>**. Notice that the scope keyword is enclosed in double quotations. Also, the scope name is case sensitive, so you have to make sure to get the case right for your scope name. An example of a query that limits on scope is:

```
SELECT rank, title, author FROM scope() WHERE CONTAINS(author, 'Tom Rizzo') AND "scope"='All Sites'
```

**WHERE Clause**

The **WHERE** clause specifies the conditions for your search. It is split into two functions. The first function for which you can use it is to group your columns together into a single alias using the **WITH..AS** predicate. This is useful for complex queries, where you may not want to specify all the columns multiple times, but would rather apply your query against the alias.

The second function of the **WHERE** clause is to evaluate your search conditions. These search conditions can be standard column value checks, **FREETEXT** and **CONTAINS** queries, or specific scopes to search. Please note that the scope must be in double quotations and is case sensitive, as discussed earlier in the **FROM** clause section.

Table 11-14 lists all the predicates you can use with SharePoint search:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AND, OR, NOT</strong></td>
<td>You can use these logical operators to compare different values in your other predicates. Please note that <strong>NOT</strong> is evaluated before <strong>AND</strong>, which is evaluated before <strong>OR</strong>. Also, you cannot use <strong>NOT</strong> and <strong>CONTAINS</strong> together as the first clause in your <strong>WHERE</strong> clause.</td>
</tr>
<tr>
<td><strong>WITH..AS</strong></td>
<td>Allows you to group multiple columns into a single alias. Only the <strong>FREETEXT</strong> predicate supports aliases created with the <strong>WITH..AS</strong> predicate. You must define the alias before using it, and the alias name must be prefixed with a # as in <strong>WITH author, title AS #coreprops</strong>.</td>
</tr>
</tbody>
</table>

(Continued)
Table 11-14 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTAINS</td>
<td>Allows for searching for terms in text columns. <strong>CONTAINS</strong> removes noise words, allows language inflection, and can search by proximity using the <strong>NEAR</strong> predicate. <strong>CONTAINS</strong> takes a column and your condition. It supports wildcard prefix searches using *, but does not support suffix wildcard searching. You can also use <strong>LCID</strong> after the condition to specify locale.</td>
</tr>
<tr>
<td>FORMSOF</td>
<td>Works with the <strong>CONTAINS</strong> predicate and allows you to find all the linguistic forms of the word, such as inflected forms (run, running, ran) or thesaurus forms (run, jog). You specify either <strong>INFLECTIONAL</strong> or <strong>THESAURUS</strong> to tell search the type of linguistic match that you want.</td>
</tr>
<tr>
<td>NEAR</td>
<td>Works with the <strong>CONTAINS</strong> predicate to allow proximity searches. You can use a shorthand form of <strong>NEAR</strong> by substituting the ~ character. When the specified words are found within approximately 50 words of each other, this predicate will return a match. The closer together the words are found, the higher the rank returned.</td>
</tr>
<tr>
<td>FREETEXT</td>
<td>Finds all the specified words throughout all your text columns. Often, you will specify all the columns in this predicate, using the <strong>DEFAULTPROPERTIES</strong> or * syntax. You should use only one <strong>FREETEXT</strong> predicate in your search queries; otherwise, relevance may be negatively impacted. You can place <strong>LCID</strong> after the search conditions to specify the locale of the search.</td>
</tr>
<tr>
<td>LIKE</td>
<td>Performs pattern matching on your properties in the property store. You can use % for zero or one matches, _ to match a single character, [ ] to specify a range such as [a-z], or [^ ] to specify ranges not to match. For example, [To^m] would match tot but not Tom. To match the literals, such as the percentage sign literal used for matching, place them in square brackets; for example, [%].</td>
</tr>
</tbody>
</table>

**Literal values**

Literals can be matched using comparison operators such as <, >, <=, >=, !=, =, or =.

**DATEADD**

The **DATEADD** function allows you to perform relative time queries, for example, within the last two hours or two days. You pass in the datetime unit, such as **YEAR**, **QUARTER**, **MONTH**, **WEEK**, **DAY**, **HOUR**, **MINUTE**, or **SECOND**. Then, you pass in the offset, which must be negative. Finally, you pass in the datetime to perform the operation on. This must be a datetime literal returned from either **GETGMTDATE** or the result of another **DATEADD** function. You can chain multiple **DATEADD** functions together.

(Continued)
Chapter 11: Building Applications with the Search API and Web Services

### Table 11-14 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multivalued ARRAY</td>
<td>Allows you to specify multiple values for comparison against a multivalued column. You pass literals in the array for comparison, in square brackets, for example [1, 2, 3]. There are quantifiers for matching, such as ALL, SOME, ANY, SOME, and ANY are interchangeable and specify that the match should be a match against any of the array elements. ALL specifies that every element on the left side of the comparison must evaluate to TRUE, when compared to the ARRAY. This predicate no longer works in SharePoint 2007.</td>
</tr>
<tr>
<td>NULL</td>
<td>Indicates whether the document has a value for the column, such as IS NULL or IS NOT NULL.</td>
</tr>
<tr>
<td>ISABOUT</td>
<td>Do not use. Deprecated but still works with the CONTAINS predicate.</td>
</tr>
<tr>
<td>RANKMETHOD</td>
<td>Do not use. Deprecated but still works with the ISABOUT predicate.</td>
</tr>
</tbody>
</table>

#### ORDER BY Clause

The ORDER BY clause lets you sort the results based on the columns and direction you specify. Ascending order is the default, if you do not specify ASC or DESC in your clause. You can also specify multiple columns. The sample below shows sorting by author and title:

ORDER BY author ASC, title DESC

### Some Example Queries

To help you understand the syntax for SQL queries, the following are some sample queries:

//CONTAINS Queries

//Simple CONTAINS
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE CONTAINS ('"bike"') AND ( ('SCOPE' = 'All Sites')) ORDER BY "Rank" DESC

//Multiple words
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE CONTAINS ('"bike helmet"') AND ( ('SCOPE' = 'All Sites')) ORDER BY "Rank" DESC

//Wildcard
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE CONTAINS (' "bik*" ')
Chapter 11: Building Applications with the Search API and Web Services

```sql
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE CONTAINS ('"bike" NEAR "helmet" ')

//FORMSOF INFLECTIONAL
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE CONTAINS ('FORMSOF ( INFLECTIONAL , "run" ) ')

//FORMSOF THESAURUS
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE CONTAINS ('FORMSOF ( THESAURUS , "Windows" ) ')

//ISABOUT
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE CONTAINS ('ISABOUT ("bike","helmet") ')

//FREETEXT Queries

//FREETEXT simple
SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE FREETEXT(DefaultProperties, 'bike')

//FREETEXT property search
SELECT Author, Title, Rank, Size, Description, Write, Path FROM scope()
WHERE FREETEXT(author, 'Jesse Merriam') AND FREETEXT(DefaultProperties, 'bike')

//OTHER Queries

//LIKE
SELECT Author, Title, Rank, Size, Description, Write, Path FROM scope() WHERE Author LIKE 'Jess %'

//LITERAL COMPARISON
SELECT Author, Title, Rank, Size, Description, Write, Path FROM scope()
WHERE Size >= 40960

//DATEADD
SELECT Title, Rank, Size, Description, Write, Path, LastModifiedTime FROM scope() WHERE LastModifiedTime <= DATEADD(DAY, -30, GETGMDATE())

//Get file by type
```
Chapter 11: Building Applications with the Search API and Web Services

```sql
SELECT Title, Rank, Size, Description, Write, Path, LastModifiedTime FROM scope() WHERE FileExtension = 'doc' OR FileExtension = 'docx'
```

```sql
// Get Folders
SELECT Title, Rank, Size, Description, Write, Path, LastModifiedTime FROM scope() WHERE ContentType = 'Folder'
```

**The Query Object Models**

SharePoint has two query object models, when you want to perform searches programmatically. Which one you use is determined by where your application will run, since one is a .NET-based class model and the other is a web service. The .NET classes cannot be made remote, so if you need to run your queries and get results from an application not running on your SharePoint Server, you will need to use the web services API. Both models pretty much perform the same functions, but the access methods will be different, since one will be called directly from .NET, and the other will be passed web service calls to get results.

The following section will outline the object models for you and explain how to use them in your applications.

**.NET Query Object Model**

You can use the .NET query object model when you are running directly on the SharePoint server, or you can make it remote, if you write your own web services or front end for a remote application that runs on the SharePoint server. The API is encapsulated in the `Microsoft.Office.Server.Search.dll` and uses the `Microsoft.Office.Server.Search.Query` namespace. There are three classes that are important inside of the namespace. The first is the base `Query` class, which you will not call directly but will leverage through the classes that inherit from it. The second is the `KeywordQuery` class, which allows you to use keyword-style queries with SharePoint. The third is the `FullTextSqlQuery` class, which allows you to use SQL syntax queries with SharePoint. Both classes use the same pattern to execute queries, and you will see that the only difference is the type and syntax of the query that you send to the object model.

**Adding a Reference to the SharePoint DLLs**

Before starting your new search application, you will want to add a reference to the `Microsoft.Office.Server.Search.dll`, and also add the namespace `Microsoft.Office.Server.Search` to your project. You will probably also want to add a reference to the `Microsoft.SharePoint.dll` and add the `Microsoft.SharePoint` namespace to your project, since you may use the SharePoint object model with your search applications. Furthermore, you will want to add `Microsoft.Office.Server` as a reference with the namespace `Microsoft.Office.Server`. To make your code easier to read, you can add the `Microsoft.Office.Server.Search.Query` namespace to the namespace directives in your code, so that you can access the query classes without using the full name.
Query Class

Before diving into some of the programming tasks that you will perform using the object model, take a look at the classes involved. The base `Query` class provides a number of properties and methods that you should understand, since the other classes derive from this base class. Table 11-15 lays out these properties and methods:

**Table 11-15**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Gets the name of the search application which is usually the GUID of the SSP.</td>
</tr>
<tr>
<td>AuthenticationType</td>
<td>Gets or sets the authentication type for the query which can be NtAuthenticatedQuery or PluggableAuthenticatedQuery.</td>
</tr>
<tr>
<td>Culture</td>
<td>The culture to use for the thesaurus, stemming, noise words, and so on.</td>
</tr>
<tr>
<td>EnableStemming</td>
<td>Boolean to turn stemming on and off.</td>
</tr>
<tr>
<td>Execute</td>
<td>Executes the query.</td>
</tr>
<tr>
<td>GetProperties</td>
<td>Returns a <code>PropertyInformation</code> array that corresponds to the properties returned for your query.</td>
</tr>
<tr>
<td>GetScopes</td>
<td>Returns a <code>ScopeInformation</code> array that corresponds to your scopes for your query.</td>
</tr>
<tr>
<td>HighlightedSentenceCount</td>
<td>Gets or sets using an integer the number of sentences to include in the hit highlighted summary. The default is 3.</td>
</tr>
<tr>
<td>Hint</td>
<td>Allows you to hint to the engine whether to optimize for the full text index or the property store for the query to get the most relevant results. This property can be <code>None</code>, <code>OptimizeWithFullTextIndex</code>, or <code>OptimizeWithPropertyStore</code>.</td>
</tr>
<tr>
<td>IgnoreAllNoiseQuery</td>
<td>Boolean that tells the engine to ignore queries that are all noise words or allow those queries to execute.</td>
</tr>
<tr>
<td>KeywordInclusion</td>
<td>Boolean that specifies whether the query results contain all or any of the specified search terms.</td>
</tr>
<tr>
<td>QueryText</td>
<td>A string that tells the engine what your query is. This is a critical property.</td>
</tr>
<tr>
<td>ResultTypes</td>
<td>This property defines the result types in your returned results, such as <code>DefinitionResults</code>, <code>HighConfidenceResults</code>, <code>None</code>, <code>RelevantResults</code>, and <code>SpecialTermResults</code>.</td>
</tr>
<tr>
<td>RowLimit</td>
<td>Gets or sets using an integer the maximum number of rows to return for your query.</td>
</tr>
</tbody>
</table>

(Continued)
Chapter 11: Building Applications with the Search API and Web Services

Table 11-15 (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartRow</td>
<td>Gets or sets the first row to include in the search result. This is used to skip initial rows in the result. This is zero based.</td>
</tr>
<tr>
<td>Timeout</td>
<td>Gets or sets in milliseconds the timeout value for your query. The default value is 10000, which is 10 seconds.</td>
</tr>
<tr>
<td>TotalRowsExactMinimum</td>
<td>Gets or sets the minimum number of rows for the query before it times out. You use this in combination with the Timeout property. The default value is 200.</td>
</tr>
<tr>
<td>TrimDuplicates</td>
<td>Boolean that tells the engine to trim or keep duplicates.</td>
</tr>
<tr>
<td>UrlZone</td>
<td>Returns a SPUrlZone, which corresponds to the URL zone that originated the search request.</td>
</tr>
</tbody>
</table>

KeywordQuery Class

The KeywordQuery class inherits from the Query class, so the properties and methods flow from the base class to this inherited class. The only added properties that are unique to this class are the EnableURLSmashing property, HiddenConstraints property, SelectProperties property, and SortList property. We'll step through these additional properties. The EnableURLSmashing property is a Boolean that turns on or off URL smashing, which is a relevancy feature. The HiddenConstraints property takes a keyword constraint such as author:Tom Rizzo. This is usually used by search alerts and RSS. The SelectProperties property returns a collection to which you can add. It is the same as if you added properties to a SELECT clause in a SQL query, in that SharePoint will return those properties. The SortList property returns a collection that you can add properties to, in the same way that you can when you add an ORDER BY clause to your SQL query.

FullTextSqlQuery Class

The FullTextSqlQuery class also inherits from the Query class. It does not add any new properties or methods to those that the base class provides.

Performing a Query and Getting Your Results

Performing a query and getting results are the same in both keyword and full-text queries. The only difference is the way that you formulate your query, since the styles of the two types of queries are different. To create your query, you need to instantiate either the KeywordQuery or the FullTextSqlQuery class. Both constructors take a SPsite, ServerContext, or a string of the SSP's GUID in order to create an instance of the object.

Once you have instantiated the object, you set your properties, formulate your query, and call the Execute method. You will need to tell SharePoint before you execute your query what types of results you want back, such as relevant results, high-confidence results, or all of the above results. The Execute method returns a ResultTableCollection object, which contains a collection of ResultTable objects that map to the ResultType enumeration for results, such as relevant results or high-confidence results. The ResultTable class implements IDataReader, so that you can access your rows in your code.
Chapter 11: Building Applications with the Search API and Web Services

I want to provide a quick note on result counts in your `ResultTable` objects. The API returns a `TotalRows` and `TotalAvailable` count for your search results. These are not exact counts, so you have to be careful when you are paging. The imprecision has to do with the way that the security trimming and search engine work. For performance reasons, SharePoint returns an approximate count, and as you page through the results, the count gets more accurate. When it comes to searching, if your users do not find what they are looking for on the first few pages, they will abandon the search. Just be aware that this count is not accurate, and this inaccuracy is something that may be addressed in a future release of SharePoint.

Table 11-6 lists the properties and methods for the `ResultTableCollection`:

**Table 11-6**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Number of <code>ResultTable</code> objects contained in the collection.</td>
</tr>
<tr>
<td>DatabaseTime</td>
<td>Time in milliseconds on how long the query took in the database.</td>
</tr>
<tr>
<td>Definition</td>
<td>Returns the <code>ResultDefinition</code> for the results.</td>
</tr>
<tr>
<td>ElapsedTime</td>
<td>The overall elapsed time for the query in milliseconds.</td>
</tr>
<tr>
<td>IgnoredNoiseWords</td>
<td>Returns the list of ignored noise words.</td>
</tr>
<tr>
<td>KeywordInformation</td>
<td>Returns a <code>KeywordInformation</code> object that provides the definition for a matched term.</td>
</tr>
<tr>
<td>QueryTerms</td>
<td>Gets or sets the terms used for the query.</td>
</tr>
<tr>
<td>SpellingSuggestion</td>
<td>Gets or sets a spelling alternative for unrecognized terms.</td>
</tr>
</tbody>
</table>

With the `ResultTableCollection`, you can retrieve the particular result set you are interested in, such as relevant results, by using the `ResultType` enumeration to index into the collection. The following code runs a keyword query, returns the results to a `DataTable`, and then displays the results in a `DataGridView`:

```csharp
KeywordQuery kq = new KeywordQuery(sc);
kq.QueryText = "Bike";

kq.ResultTypes = ResultType.RelevantResults | ResultType.HighConfidenceResults;

kq.QueryText = "bike";

// If you want to add new properties, shown below
//kq.SelectProperties.Add("Title");
//kq.SelectProperties.Add("Author");

ResultTableCollection rt = kq.Execute();

ResultTable resultstable = rt[ResultType.RelevantResults];
```
Chapter 11: Building Applications with the Search API and Web Services

DataTable resultsdt = new DataTable();
resultsdt.Load(resultstable, LoadOption.OverwriteChanges);
dataGridView1.DataSource = resultsdt;

One thing to point out with hit highlighting is that, if you want to display the hit highlighted words, you will have to parse the returned results and search for tags (such as c0 or c1), which surround the hit highlighted words in your search results.

The following code shows the same query, but this time as a full-text query, not a keyword query:

ServerContext sc = ServerContext.GetContext(spsite);

FullTextSqlQuery ft = new FullTextSqlQuery(sc);

ft.QueryText = "SELECT Rank, Title, Author, Size, Path, Description, Write, SiteName, CollapsingStatus, HitHighlightedSummary, HitHighlightedProperties, ContentClass, IsDocument FROM Scope() WHERE FREETEXT(DEFAULTPROPERTIES, 'Bike')";

ft.ResultTypes = ResultType.RelevantResults | ResultType.HighConfidenceResults;
ResultTableCollection rt = ft.Execute();

ResultTable resultstable = rt[ResultType.RelevantResults];
DataTable resultsdt = new DataTable();
resultsdt.Load(resultstable, LoadOption.OverwriteChanges);
dataGridView1.DataSource = resultsdt;

Since these are DataTable objects being returned, you could even do more with your search results, such as use LINQ to refine the results even further. It is up to you how far you go with the results, since they do use standard ADO.NET constructs.

What about Federation?
As you can see, federated results are not an option in the ResultType enumeration as one of the result types to return in your queries. The only extensibility for federation in SharePoint today is that you can inherit from the federated web parts and extend those web parts. From a query object model standpoint, you cannot issue or consume data from the default federation technology in SharePoint.

Web Service Query Object Model
SharePoint also provides a Query web service that you can work with to perform your remote queries and get your results. The web service builds on the standards, such as Web Services Description Language (WSDL), so that you can use any development tool or environment that supports web services...
to write for the SharePoint search service. We will see later in the chapter how ASP.NET server controls can call the web service and can add Search to any ASP.NET application.

Connecting to the Web Service and Creating a Proxy

The web service definition is contained in the search.asmx file which is located in the _vti_bin directory for your SharePoint site. Therefore, the full URL to connect to the web services would be http://server/_vti_bin/search.asmx. Visual Studio makes it easy to add a web service as a reference and will automatically create the proxy for you. Just add the SharePoint search web service as a web reference in Visual Studio, as shown in Figure 11-2.

![Figure 11-2: Adding the Search Web Service in Visual Studio](image)

Please note that if you are using only WSS, you will instead use the spsearch.asmx file for your web service reference.

Performing Your Query and Getting Results

To work with the QueryService, you will use the methods of the Query web service and pass XML constructs back and forth to perform the query and get your results. In code, you need to instantiate the QueryService of SharePoint before you can start writing to the web service. This is done by assigning a QueryService object to a new instance of the QueryService object from the web service, as shown in the code that follows:

```csharp
SearchWS.QueryService query = new SearchWS.QueryService();

//Set the credentials to talk to the web service to the credentials of the logged in user.
query.UseDefaultCredentials = true;
```
Once you have instantiated your object, you can start programming against the service. One thing to remember is that you will want to set your credentials correctly when working with the web service. Normally, you will want to pass the credentials of the current user. The way to do this is to set the UseDefaultCredentials properties on your QueryService object to be true. Remember to qualify your reference to your web service proxy, if you do not see the UseDefaultCredentials property in the IntelliSense for Visual Studio, or if you get an error compiling your code.

To send your queries, you need to create a query packet, which is an XML-formatted document that contains all the settings that you want for your query, such as whether the query is a keyword or full-text query, and the query terms. The SDK does a good job of covering the packet format, so we will not spend time going over all the properties you can set in the packet, since most of the time you will just set the type of query and the terms.

Query Method

The Query method is straightforward in that you pass in your XML packet, and you get back your result as an XML string that you can parse. One thing to note is that you can use Keyword or SQL Fulltext format for your queries. If you use the Keyword syntax, you will not be able to specify the columns returned in your result or what scopes to use. The following code shows how to send a query to SharePoint using the Query method:

```csharp
private StringBuilder KeyWordQueryPacketBuilder(string QueryText)
{
    StringBuilder sb = new StringBuilder();
    // The QueryText is set to "STRING" for keyword queries
    sb.Append("<QueryPacket xmlns="urn:Microsoft.Search.Query">");
    sb.Append("<Query>");
    sb.Append("<SupportedFormats>");
    sb.Append("</SupportedFormats>");
    sb.Append("<Context>");
    sb.Append("<QueryText type="STRING" language="en-us">" + QueryText + "</QueryText>");
    sb.Append("</Context>");
    sb.Append("</Query>");
    sb.Append("</QueryPacket>");
    return sb;
}
```

//Main section of code
StringBuilder sb = new StringBuilder();
//Send a keyword query
sb = KeyWordQueryPacketBuilder("bike");

//Perform a Query
try
Chapter 11: Building Applications with the Search API and Web Services

```csharp
{ 
    string results = query.Query(sb.ToString());
    MessageBox.Show(results);
} 
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

**QueryEx Method**

The main differences between the `Query` and `QueryEx` methods are that the `QueryEx` method returns its results as a dataset rather than just an XML document. Also, with the `QueryEx` method, you can get different result types, such as relevant, highConfidence, or definition results, which you cannot do with the `Query` method, since it only will return relevant results. The way you get these extra results is by customizing the query packet to include the right elements, such as `<IncludeSpecialTermResults>` or `<ImplicitAndBehavior>`, in your XML. Once you get back your results as a dataset, you can work with that dataset just like any other ADO.NET dataset. The following code makes a query and returns the dataset to a `DataGridView` object:

```csharp
private StringBuilder SQLQueryPacketBuilder(string QueryText)
{
    StringBuilder sb = new StringBuilder();
    // The QueryText is set to "MSSQLFT" for SQL queries.
    sb.Append("<QueryPacket xmlns="urn:Microsoft.Search.Query"> ");
    sb.Append("<Query>");
    sb.Append("<SupportedFormats>");
    sb.Append("<QueryText type="MSSQLFT" language="en-us">" + QueryText + "</QueryText>");
    sb.Append("<Context>");
    sb.Append("<QueryPacket>");
    return sb;
}

//Main section of code
sb = SQLQueryPacketBuilder("<![CDATA[ SELECT Title, Rank, Size, Description, Write, Path FROM scope() WHERE FREETEXT(DefaultProperties, 'bike') AND (("SCOPE" = 'All Sites')) ORDER BY "Rank" DESC ]]>");

//Perform a QueryEx query
try
{
    DataSet resultSet = query.QueryEx(sb.ToString());
    dataGridView1.DataSource = resultSet.Tables[0];
} 
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
```
Registration Method

With registration, a client application can register with the SharePoint web service. You have two actions you can perform, which are creating a new registration or updating an existing registration. Most times registration is used by the research pane in Office or Internet Explorer. The following code registers with the SharePoint Search service. On successful calls, you will get back a string that describes the service to you, such as the query path to send queries; the type of service, such as SOAP; and the types of services that the search provider supports.

```xml
<ProviderUpdate
xmlns="urn:Microsoft.Search.Registration.Response"><Status>SUCCESS</Status>
<Providers><Provider><Id>{89ad732c-177b-481b-aaf2-12a13d4dea54}</Id><Name>Microsoft Office SharePoint Server 2007</Name><QueryPath>http://moss/_vti_bin/search.asmx</QueryPath><Type>SOAP</Type><Services><Service><Id>{89ad732c-177b-481b-aaf2-12a13d4dea54}</Id><Name>Search Center</Name><Category>INTRANET_GENERAL</Category><Description>This service allows you to search the site : Search Center</Description><Copyright>Microsoft® Office SharePoint® Server 2007 Search</Copyright><Display>On</Display></Service></Services></Provider></Providers>
```

GetPortalSearchInfo Method

This method returns all the scopes as a string of XML data:

```xml
<SiteConfigInfo xmlns="urn:Microsoft.MSSearch.Response.Config"><Name>Search Center</Name><Id>{89ad732c-177b-481b-aaf2-12a13d4dea54}</Id><Scopes><Scope><Name>People</Name></Scope><Scope><Name>All Sites</Name></Scope><Scope><Name>Global Query Exclusion</Name></Scope><Scope><Name>Rank Demoted Sites</Name></Scope><Scope><Name>Author is Jesse</Name></Scope></Scopes></SiteConfigInfo>
```

GetSearchMetadata Method

The GetSearchMetadata method returns an ADO.NET dataset that contains two tables. The first table is all the managed properties for search, their name, type, and whether they are retrievable, and queryable. The second table is all the scopes for search with the name and description of the scopes.

Status Method

The Status method returns a string, which is the current status for the service, such as ONLINE or OFFLINE.

RecordClick Method

Even though there is a RecordClick method that allows you to record your search usage, there is a bug with the method that prevents you from using the method as intended. We hope that this will be fixed in the future, since API calls are not recorded as search queries in your query logs for your overall Search usage reporting.
Microsoft Office Research Pane Integration

Microsoft Office also supports the SharePoint Search service web services using the research pane. You can perform SharePoint searches from within Office, just like you can perform spelling or thesaurus lookups. To add the SharePoint Search service, all you need to do is launch your research pane, under the research options, add a new service, and point that service at the SharePoint web service, as shown in Figure 11-3.

Figure 11-3: Adding SharePoint in Research Services

Once you are done with that, you can perform a search from the research pane against your SharePoint environment, as shown in Figure 11-4.
Chapter 11: Building Applications with the Search API and Web Services

Gotchas When Using the Object Models

There are three things that neither the web service nor the .NET object models implement that are implemented by the Search Center and its web parts. When you use the object model, there is no query logging, so your queries will not be recorded in the query log, which can lead to incorrect usage reporting. Also, you do not get federated results. Finally, hit highlighting has to be done manually. You get the results but not the formatted highlighting.

Tools to Help You Develop Search Applications

There are a number of tools to help you get started with building search applications. The set of tools we highly recommend are all contained in the Search Community Toolkit on Codeplex, http://www.codeplex.com/sct. It contains everything from ASP.NET server controls to add Search to your applications to .NET helper classes for the Search object models.

Of all the tools in the community kit, this tool will be the one you find yourself running the most. It is written in VB.NET and provides the ability for you to navigate your search environment by showing you all the scopes and managed properties you have. It also provides the ability to point and click to create your keyword or SQL queries, to display the query packets that will be sent to the server, and to browse the results that come back. It is a highly recommended tool. Figure 11-5 shows the tool in action.
Content Query Web Part and Cross-List Queries

Even though we will not be covering this in detail, you should be aware of the Content Query web part and the cross-list capabilities of SharePoint. They work together to provide the capability to search across many different lists in your SharePoint environment and visualize the results. While this does not use the search engine included in SharePoint, both of these technologies are search technologies, and are something you should consider using to build your solutions. You should use these technologies when searching data contained within a single site collection within SharePoint. The queries and web part do not span site collections or different datasources.

Conclusion

In this chapter you saw how to administer your SharePoint environment and perform searches against your SharePoint environment. The object model and web service are powerful tools that allow you to work with and customize SharePoint Search in new ways. By leveraging the APIs, you can incorporate Search easily into your existing applications or write a new class of application that harnesses the power of SharePoint Search.
A

**AccessChecker element, BDC metadata model**, 104–105, 202–203

**ACLs (access control lists)**
caching security permissions and, 194custom security trimmers as alternative to, 191SharePoint security and, 189–190

**Actions, Action, and ActionParameter elements**in BDC metadata model, 101working with, 128–129

**Active Directory Federation Services (ADFS)**, 231

**AD (Active Directory) database**BDC supplemental properties, 142–143people information stored in, 131profile imports and, 134–138

**Add Colleagues web page**, 149

**Add Mapping user interface**, 241–242

**Add to My Colleagues link**, 148

**ADF (application definition file)**BDC and, 87configuring BDC import, 143–145importing from LOB applications, 134

**ADFS (Active Directory Federation Services)**, 231

**Administration, Search Server 2008, 16–18 administration object model**content management. See content managementcrawl rules, 300–302Federated Locations, 297–300file types, 302

**alerts, search**, 72

**algorithms**extraction algorithm for document titles and authors, 282ranking elements and, 254search algorithms, 2–3

**All Sites tab, SCT (Search Center with Tabs)**, 146

**Always trigger, federated searches**, 164

**anchor text**relevance tuning and, 259–260search algorithms and, 3

**anonymous credential, federated searches**, 178

**APIs (application programming interfaces)**BDC API, 122ISecurityTrimmer API, 191–192

**Appearance settings, Federated Search Results web parts**, 187

**application definition file. See ADF (application definition file)**

**application programming interfaces (APIs)**BDC API, 122ISecurityTrimmer API, 191–192

**keywords**, 308–309


**Advanced Search Options pane**, 151

**Advanced Search page**, 272

**Advanced Search Results web part**, 244–245

**alerts**, search, 72

**administration**, Search Server 2008, 16–18administration object modelcontent management. See content management
applications, for searches. See search applications
architecture
  BDC, 85–86
  SharePoint security, 189–190
ASPX files, 116, 146–147
Associations/Association elements
  in BDC metadata model, 108–109
  working with, 128–129
authentication
  BDC security trimming, 201
  configuring Search Server for, 179–180
  configuring SharePoint Server for, 180
  crawling basic authentication site, 68–69
  FBA (Forms Based Authentication), 191
  User Federated Location and, 179
authoritative pages
  click distance from, 72
  ranking algorithms, 280–282
authoritative sites, working with, 310–311
authors, extraction algorithm for, 282
autodiscovery, of term definitions, 231
Automatic Definitions, 273

B
backing up searches, 41–42
basic authentication
  crawling basic authentication site, 68–69
  User Federated Locations and, 179
BDC (Business Data Catalog)
  AccessChecker element, 104–105
  Actions, Action, and
    ActionParameter elements, 101, 128–129
  as API, 122
  application definition file, 87
  architecture and benefits of, 85–86
  assemblies, 122–123
  Associations and Association elements, 108–109, 128–129
  connecting to Shared Services database, 125
  content sources and, 65, 117–118
  custom applications for, 123–124
  Definition Editor tool, 87–88
  Entities and Entity elements, 94–95, 125–126
  executing methods and displaying results, 126–128
  FilterDescriptors and FilterDescriptor elements, 98–101
  filters, 109
  Finder and SpecificFinder methods, 102–104
  GenericInvoker element, 106
  Identifiers and Identifier elements, 95–96
  IEnumerable element, 104–105
  lists, 109, 113–115
  LobSystem, 89–91
  LobSystemInstances and
    LobSystemInstance, 92–94, 125
  mapping crawled properties, 119
  metadata model for, 88–89
  MetadataObject base class, 89
  MethodInstance element, 101–102
  Methods and Method elements, 96–98
  modifying profile page, 116–117
  namespaces, 123
  overview of, 85
  Parameters and Parameter elements, 98
  Scalar element, 106
  scope of searches, 119–120
  searching, 117
  security trimming. See security trimming, BDC
  SharePoint Designer and, 120–122
  troubleshooting, 129
  TypeDescriptors, TypeDescriptor, DefaultValue elements, 106–108
  ViewAccessor element, 104
  web parts, 109–112
  WSS search and, 15
  XML samples, 109
  XSD schema file, 87
BDC Definition Editor tool, 87–88
BDC MetaMan utility, 87
best bets
relevance and, 274–276
security trimming and, 191
working with, 309–310
**Best Bets** web part, 191
**Booleans**
  keyword syntax, 313
  SQL syntax in search applications, 315
**branding icons, in federated searches, 174**
breakpoints, in debugging, 200
**Business Data Actions** web part, 112
**Business Data Catalog.** See **BDC (Business Data Catalog)**
**Business Data Item Builder** web part, 112
**Business Data Item** web part, 111–112
**Business Data List** web part, 110
**Business Data Picker,** 114–115
**Business Data Related List** web part,
  110–111
**BusinessDataContentSource**, 293–294

**C**

**caching**
  location settings, 187
  security permissions, 194
**case-sensitivity**
  in searches, 82
  thesaurus and, 224
  XML and, 90
**categories, query, 305**
**CategoryCollection** object, 305
**Central Administration, 45**
**central processing units (CPUs), 35–37**
**CheckAccess method**
  BDC security trimming and, 202–203
  ISecurityTrimmer methods, 192, 195–196
**click distance**
  authoritative and nonauthoritative pages and, 280–282
  content relevance calculated by, 72
  search algorithms and, 3
**Codeplex, Search Community Toolkit,**
  251–252, 330
**colleagues**
  social networking and, 148–149
  working with, 225
**common credential, federated searches, 178**
**Communication Server, Microsoft, 148**
**comparison filters, for searching LOB systems, 100**
**configuring/administering searches, 45–83**
  alerts, 72
  authoritative pages and, 72
  case-sensitivity and diacritic-sensitivity, 82
  Central Administration, 45
  content databases and, 57–58
  content sources, 60–62, 64–66
  crawl logs, 70–71
  crawl rules, 66–70
  crawl size settings, 82
  federated searches, 73
  file types, 72
  filter pack and, 80–81
  full vs. incremental crawls, 62–63
  IFilters, 80
  managed properties, 73–75
  Office SharePoint server search service,
    46–50
  reports, 78–79
  resetting crawled content, 72
  resetting search services, 82
  result removal, 78
  scheduling, 63–64
  scope, 75–77
  search service management, 52–57
  search services, 45–46
  server name mappings, 77
  SSP settings, 59–60
  WSS search, 50–52
**connection sources, Profile Services, 132**
**connections, Profile Services, 133–134**
**connectors**
  for Documentum and FileNet, 23
  indexing, 155
**content**
  additional sources, 64–65
  avoiding double indexing content, 159
  crawling BDC data, 117–118
content (continued)

default sources, 60–62
federated searches for content outside firewall, 159
hardware requirements for searching, 35–36
interacting with content source, 65–66
types of, 1–2
Content Access account, 212
Content class, 291–292
content databases, 57–58
content management
Content class, 291–292
content sources and, 291
crawling content sources, 295–297
creating content sources, 293–295
deleting content sources, 295
enumerating content sources, 295
relevance and, 255–258
Content Query web part
search applications and, 312
searching across lists, 331
content sources
additional, 64–65
crawling, 295–297
creating, 293–295
default, 60–62
deleting, 295
enumerating, 295
interacting with, 65–66
overview of, 291
relevance-related settings, 256–258
cookie authentication
custom security trimmers as alternative to, 191
User Federated Locations and, 179
Core Search Results web part
customizing properties for, 219
inheriting from, 251
modifying, 243–244
turning on stemming, 224
counters, performance monitoring, 39
CPUs (central processing units), 35–37
crawl logs, 70–71, 311
crawl rules
associating security trimmers with, 193
basic authentication and, 68–69
creating, 300–302
creating and attaching to security trimmer, 198–199
excluding portion of Web site, 69–70
overview of, 56–57, 66–67
relevance tuning with, 258
crawl schedules
overview of, 63–64
relevance tuning and, 256–258
crawl settings control, 256–258
crawled properties
administering, 238–241
mapping, 119
queries, 305
crawling. See also indexing
basic authentication site, 68–69
BDC data, 117–118
content sources, 295–297
default sources, 60–62
federated searches compared with, 158–159
full vs. incremental, 62–63
logs, 70–71, 311
resetting crawled content, 72
scheduling, 63–64, 256–258
security and, 191
size settings, 82
subdirectories, 293
CrawlRule class, 300–301
credentials
BDC authentication and, 201
federated searches, 177–180
cross-list capabilities, 331
custom searches. See searches, custom
CustomContentSource, 293

data, freshness of searched data, 158–159
data types, SQL, 315
Data View web part, 231–234
database servers, planning/deploying, 29
databases
managing content databases, 57–58
storage and disk optimization of search property store database, 37–38
storing people information in, 131
dates, SQL syntax in search applications, 315
debugging
  BDC security trimmer, 209–212
  custom security trimmer, 200
Default Content Access account, securing
  Search Server 2008, 212
DefaultValue element, BDC metadata model, 106–108
deploying
  BDC security trimmer, 206–209
  Enterprise Searches. See Enterprise Search deployment
diacritics
  diacritic-sensitivity in searches, 82
  relevance and, 267–268
dictionaries, custom, 268
  Did you mean? feature, 273
digest authentication, 179
DIP (Document Information Panel), 114–115
disk space requirements, for Enterprise
  searches, 36–37
Display Information section, federated searches, 172–173
Display Size template, XSLT, 230–231
Display String template, XSLT, 231
DLLs, 320. See also .NET assemblies
Document Collapsing template, XSLT, 231
Document Information Panel (DIP), 114–115
documents
  IRM-protected, 191
  Office documents as content type, 1–2
  property weighting and, 282
Documentum, connectors for, 23, 155
duplicate collapsing, 271–272
dynamic ranking elements, 254

E
Enterprise Search deployment
  database servers, 37–38
  hardware selection, 34–35
  index search recovery options, 42–43
  index server, 35–37
  key components, 27–29
  large farm environment, 32–33
  medium farm environment, 31–32
  overview of, 27
  performance monitoring, 39–41
  query servers, 37
  search backups, 41–42
  Search Federation for scaling searches, 43
  single server environment, 29–30
  small farm environment, 30
  software boundaries, 33–34
  testing, 38
  three-server farm environment, 31
  topology options, 29
Enterprise Search, introduction to
  content types and, 1–2
  reasons for, 1
  search algorithms and, 2–3
  security and, 2
  Web protocols and, 3–4
Enterprise Search products
  connectors for Documentum and
    FileNet, 23
  Exchange Server and SQL Server, 25
  FAST (Fast Technology and Transfer), 25
  Microsoft filter pack, 23
  Microsoft Office SharePoint Server, 20–23
  Microsoft Search Server 2008, 16–20
  overview of, 5
  Windows Desktop Search (WDS), 5–6
  Windows Live, 24
  Windows Vista Search, 6–11
  WSS search, 11–16
Enterprise Search security. See security;
  security trimming
Entities/Entity elements
  in BDC metadata model, 94–95
  working with, 125–126
Exchange, connectors for, 155
Exchange Server, full-text search capabilities of, 25
ExchangePublicFolderContentSource, 293
expansion sets, synonyms and, 266
Extensible Markup Language. See XML
  (Extensible Markup Language)
Extensible Schema Definition (XSD), 87
Extensible Style Language Transformation

**Extensible Style Language Transformation.**
See XSLT (Extensible Style Language Transformation)

**Extensible Style Language (XSL)**
hit highlighting with, 271
writing custom XSL, 175

**Extension class, 302**
**ExtensionCollection class, 302**

to local index, 171–172
Location Type option, 168
locations, 160–163
More Results link, 172
named capture groups, 165–168
OpenSearch location, 168–170
overview of, 73, 155–157
properties for returning result metadata, 175–176
query templates, 168
required and optional query template
tokens, 170–171
restrictions, 177
results, 324
Sample Data sections, 176
Search Server 2008 and, 18–19
security trimming, 180–181
Top Federated Results web part, 187–188
triggers, 163–165
unified results sets from, 158
writing custom XSL, 175
WSS search and, 16

**federation, defined, 156**

**Federation web parts**
Federated Search Results web part, 183–185
Federated Search Results web part
top properties, 185–187
overview of, 182–183
Top Federated Results web part, 187–188

**file type biasing, search algorithms, 3**

**file types**
search administration and, 72
working with, 302

**FileNet, connectors for, 23, 155**

**FileStreamContent Source, 293**

**filter pack**
installing, 80–81
Microsoft search architecture and, 23

**FilterDescriptors/FilterDescriptor elements, BDC metadata model, 98–101**

**filters**
BDC (Business Data Catalog), 109
filter pack for Microsoft search
architecture, 23, 80–81

**FAST (Fast Technology and Transfer), 25**

**FBA (Forms Based Authentication), 191**

**Federated Location Definition (FLD) files, 156**

**Federated Locations**
administration object model, 297–300
defined, 156
fields, 161–163
Location Type option, 168
overview of, 160
preconfigured, 160
Query Template section, 168–170
triggers and, 163–165
working with, 297–300

**Federated Search Results web part**
customizing title of, 174–175
overview of, 183–185
properties, 185–187

**federated searches**
avoiding double indexing content, 159
branding icons and, 174
connectors for, 157
for content outside firewall, 159
crawling and indexing compared with, 158
credentials and, 177–180
customizing title of federated results web
part, 174–175
Display Information section, 172–173
Federated Search Results web part, 183–185
Federated Search Results web part
properties, 185–187
Federation web parts, 182–183
freshness of data and, 158–159

**F**

**file type biasing, search algorithms, 3**

**file types**
search administration and, 72
working with, 302

**FileNet, connectors for, 23, 155**

**FileStreamContent Source, 293**

**filter pack**
installing, 80–81
Microsoft search architecture and, 23

**FilterDescriptors/FilterDescriptor elements, BDC metadata model, 98–101**

**filters**
BDC (Business Data Catalog), 109
filter pack for Microsoft search
architecture, 23, 80–81
Index

property filters, 313–314
property weighting compared with property filters, 283
Finder/SpecificFinder methods
BDC metadata model, 102–104
working with BDC entities, 125–126
firewalls, federated searches for content outside, 159
fixed queries, 251
FLD (Federated Location Definition) files, 156
Form authentication, User Federated Locations and, 179
Forms Based Authentication (FBA), 191
FROM clause, SQL syntax in search applications, 316
full-text searches, Exchange Server and SQL Server, 25
FullTextSqlQuery class, 322

G
GAC (Global Assembly Cache), 198
gacutil utility, 198
GenericInvoker element, in BDC metadata model, 106
GetPortalSearchInfo method, in Web service query object model, 328
GetSearchMetadata method, in Web service query object model, 328
GetSecurityDescriptor method, 190
Global Assembly Cache (GAC), 198
global ranking parameters, 284–287
Globally Unique Identifier (GUID), for SSPs, 222–223
group policies, Windows Vista Search, 9
GUID (Globally Unique Identifier), for SSPs, 222–223

H
HA (High Availability)
index and query roles and, 46
SharePoint search environments and, 30–33
hardware selection, for SharePoint search, 34–35
hexadecimal numbers, SQL syntax in search applications, 315
High Availability (HA)
index and query roles and, 46
SharePoint search environments and, 30–33
high-confidence matches, 273–274
results, 276–279
Hit Highlighting template, XSLT
customizing, 235–237
overview of, 230
relevance and, 271
HTML (Hypertext Markup Language)
Enterprise Search and, 3
transforming XML into, 175
HTTP (Hypertext Transfer Protocol), 3
icons, branding icons in federated searches, 174
Identifiers/Identifier elements, in BDC metadata model, 95–96
ID Enumerator element, in BDC metadata model, 104–105
IFilters
filter pack for Microsoft search architecture, 23
overview of, 80
WSS search and, 15
IM (instant messaging)
presence information indicator and, 148
SharePoint colleagues and, 225
Impersonation/Delegation, BDC authentication, 201
imports
configuring BDC import, 143–145
configuring profile imports, 134–138
profile imports, 132
inclusion/exclusion, keyword syntax, 313
index role
high availability and, 46
planning/deploying, 27–28
index servers

index servers
  creating, 48–50
  hardware selection and, 35–37
  recovery options, 42–43
  scaling, 43
  SSP and, 28
indexing. See also crawling
  controlling for security, 191
  federated searches compared with, 158–159
  federating to local index, 171–172
  language detection during, 260–261
  propagating indexes to query servers, 312
Information Rights Management (IRM), 191
inheritance, from Core Search Results web part, 251
Initialize method, ISecurityTrimmer class, 192
instant messaging (IM)
  presence information indicator and, 148
  SharePoint colleagues and, 225
Instant Search, Windows Vista Search, 9–11
Internet Search
  Enterprise Search compared with, 1–4
  Windows Live and, 24
IRM (Information Rights Management), 191
ISecurityTrimmer API
  implementing custom security trimmer, 192–193
  interface, 195
  methods, 195–196
  overview of, 191–192

K

Kerberos
  delegation, 201
  User Federated Locations and, 179
Keyword class
  properties and methods, 308
  working with, 308–309
Keyword object, 308
KeywordQuery class, 322
keywords
  relevance and, 274–276
  syntax in search applications, 313–314
  working with, 308–309

L

language
  custom dictionaries, 268
  detection at index time, 260–261
  detection at query time, 262
  diacritics, 267–268
  hardware requirements for searching, 36
  noise words, 264–265
  search algorithms for detecting, 3
  thesaurus, 265–267
  word breakers and stemmers and, 262–264
large farm environment, 32–33
LastIdSeen filter, searching LOB systems, 100–101
LDAP (Lightweight Directory Access Protocol)
  BDC supplemental properties, 142–143
  creating LDAP connections, 137
  storing people information in, 131
Length Normalization setting, relevance tuning and, 283–284
letters, Pattern trigger for matching, 165
limit filters, searching LOB systems, 100
line-of-business systems
  See LOB (line-of-business) systems
lists
  BDC, 109
  cross-list capabilities, 331
  search queries and results, 220
  SharePoint, 113–115
LOB (line-of-business) systems
  importing ADF from LOB application, 134
  searching, 99–101
  trimming search results from, 202
LobSystem, in BDC metadata model, 89–91
LobSystemInstances/LobSystemInstanceId
Index

displaying, 125
Federated Search Results web parts, 185–186
Top Federated Results web part, 187–188
Location Type option, Federated Locations, 168
LocationConfiguration object, 298–299
LocationConfigurationCollection object, 297–298
locations, federated. See Federated Locations
Log Viewer (application), downloading, 209–210
Log Viewer, in administration object model, 311
logs
crawl logs, 70–71, 311
query logs, 269–270
ULS (Unified Logging Service), 209–210
viewing search logs, 311
LogViewer object, 311
Lotus Notes, connectors for, 155
LotusNotesContentSource, 293

M
managed properties
adding new, 241–243
creating/mapping new, 306–307
queries, 306
searching against, 73–75
mapping
Add Mapping user interface, 241–242
crawled properties, 119
managed properties, 306–307
metadata properties, 238
server names, 77
medium farm environment, 31–32
metadata
content sources and, 73
managing, 258–259
properties for returning result metadata in federated searches, 175–176
property mappings, 238
metadata, in BDC model
AccessChecker element, 104–105
Actions, Action, and ActionParameter elements, 101
Associations/Association elements, 108–109
Entities/Entity elements, 94–95
FilterDescriptors/
   FilterDescriptor elements, 98–101
Finder/SpecificFinder methods, 102–104
GenericInvoker element, 106
Identifiers/Identifier elements, 95–96
IDEnumerator element, 104–105
LobSystem, 89–91
LobSystemInstances/
   LobSystemInstance, 92–94
MetadataObject base class, 89
MethodInstance element, 101–102
Methods/Method elements, 96–98
overview of, 88–89
Parameters/Parameter elements, 98
Scalar element, 106
TypeDescriptors, TypeDescriptor,
   DefaultValue elements, 106–108
ViewAccessor element, 104
MetadataObject base class, in BDC metadata model, 89
MethodInstance element, in BDC metadata model, 101–102
methods
Content class, 292
CrawlRule class, 300–301
Keyword class, 308
Keyword object, 308
LocationConfiguration object, 298–299
LocationConfigurationCollection object, 297–298
Query class, 320–321
ResultTableCollection, 323
Schema object, 305
Scopes class, 302–303
Methods/Method elements

**Methods/Method elements**
- in BDC metadata model, 96–98
- executing methods and displaying results, 126–128

**metrics, relevance**, 253

**Microsoft Communication Server**, 148
**Microsoft Exchange**, connectors for, 155

**Microsoft Office**
- documents, 1–2
- support for SharePoint Search service, 329–330

**Microsoft Office SharePoint Server Search, 20–23**
- configuring/administering search services, 46–47
- connectors for Documentum and FileNet, 23
- creating index server, 48–50
- creating query server, 47–48
- data searches with BDC, 21–23
- overview of, 20
- people searches, 20–21

**Microsoft Outlook 2007**
- SharePoint colleagues and, 225
- Windows Vista Search, 6–7

**Microsoft Search Server 2008, 16–20**
- anchor text and, 260
- authentication, 179–180
- connectors for Documentum and FileNet, 23
- diacritic support, 268
- editions of, 19–20
- extending WSS search capabilities, 20
- feature set, 155
- federated searches. See federated searches
- federation capabilities of, 18–19
- noise words, 264
- overview of, 16
- ranking elements, 254
- security, 212–213
- setup and administration, 16–18
- user interface, 270
- word breakers, 261

**Microsoft.Office.Server**
- adding references to VS projects, 289
- administration names, 290
- BDC namespaces in, 123
- BDC .NET assemblies, 122
- Microsoft.SharePoint.Portal.dll, 122
- More Results for Fixed Query template, XSLT, 231
- More Results link, federated searches, 172
- My Site technology, 225

**N**
- named capture groups, federated searches, 165–168
- namespaces, BDC, 123
- .NET assemblies
  - BDC classes, 122–123
  - custom security trimmers as, 192
  - signing and adding to GAC, 198
- .NET query object model, 320–324
  - adding references to SharePoint DLLs, 320
  - classes, 321–322
  - overview of, 320
  - performing query, 322–324
- No Search Results template, XSLT, 229–230
- noise words, 264–265
- nonauthoritative pages, ranking algorithms, 280–282
- NTFS file shares, connectors for, 155
- NTLM authentication, 179
- numbers
  - Pattern trigger for matching, 164–165
  - SQL syntax in search applications, 315

**O**
- object models
  - administration. See administration object model
  - queries. See query object models

**Office**
- documents, 1–2
- support for SharePoint Search service, 329–330
Office SharePoint Server Search. See Microsoft Office SharePoint Server Search
Okapi BM25 algorithm, 254
OpenSearch
  Enterprise Search and, 4
  Federated Locations and, 156
  location of query template for federated services, 168–170
  security trimming and, 181
ORDER BY clause, SQL syntax in search applications, 316–318
Outlook 2007
  SharePoint colleagues and, 225
  Windows Vista Search, 6–7

P
parameters
  global ranking parameters, 284–287
  XSLT for transforming web part parameters, 228
Parameters/Parameter elements, in BDC metadata model, 98
password filter, searching LOB systems, 99
Pattern trigger
  federated searches, 164–165
  regular expressions used with, 165–166
People Search Box web part, 225, 246–250
People Search Core Results web part, 152, 225
people searches
  customizing, 150–154
  customizing results, 225
  customizing using properties, 246–250
  overview of, 145
  results page, 147–149
  Search Center and, 145
  search page and tabs for, 146–147
People tab, SCT (Search Center with Tabs), 146
performance
  BDC security trimming and, 202
  security trimming and, 193
performance monitoring, Enterprise searches and, 39–41
permissions, caching security
  permissions, 194
platform services, WSS search, 14–16
POC (proof of concept), testing hardware and, 38
policies
  IRM-protected documents, 191
  Profile Services, 132
predicates, SQL query syntax, 316–317
Prefix trigger, federated searches, 164–165
presence information, indicator for, 148
products, for Enterprise Search. See Enterprise Search products
profile pages, modifying, 116–117
Profile Services
  connections, 133–134
  profile imports and, 132
proof of concept (POC), testing hardware and, 38
propagation, of indexes to query servers, 312
propagation class, 312
properties. See also managed properties
  Content class, 292
  CrawlRule class, 300–301
  custom properties in searches, 237–241
  Federated Search Results web part, 185–187
  filtering by, 313–314
  Keyword class, 308
  Keyword object, 308
  LocationConfiguration object, 298–299
  LocationConfigurationCollection object, 297–298
  people searches and results and, 246–250
  Query class, 320–321
  ResultTableCollection, 323
  Schema object, 305
  Scopes class, 302–303
  web parts, 225
properties, user profiles
  BDC supplemental, 142–143
  configuring, 139–142
  overview of, 132, 138–139
property length normalization, search algorithms, 3
property picker, 245
property store database, 37–38
property weighting
  changing, 307
  ranking algorithms, 282–284
  search algorithms, 3
protocol handlers
  connectors for Documentum and FileNet, 23
  WSS search and, 15
proxy servers
  creating, 325
  managing search services, 55

Q
queries
  building, 312
  categories, 305
  crawled properties, 305
  Did you mean? feature, 273
  examples, 318–320
  fixed, 251
  language detection during, 262
  managed properties, 306
  performing .NET query, 322–324
  performing web service query, 325–326
  processing, 215
  reports, 78–79
Query class, 320–321
query logs, 269–270
Query method, Web service query object model, 326–327
query object models
  gotchas when using, 330
  .NET query object model. See .NET query object model
  overview of, 320
  Web service query object model. See Web service query object model
  WSS search and, 15
query role
  high availability and, 46
  planning/deploying, 28
  query servers and, 37
query servers
  creating, 47–48
  propagation of indexes to, 312
  query role and, 37
query templates, federated searches
  OpenSearch location, 168–170
  overview of, 168
  required and optional, 170–171
QueryEx method, Web service query object model, 327

R
RAID (redundant array of inexpensive disks), 37
RAM (random access memory), 36
ranking
  built-in ranking elements, 254
  relevance compared with, 253
  working with, 310–311
ranking algorithms
  authoritative and nonauthoritative pages, 280–282
  global ranking parameters, 284–287
  overview of, 280
  property weighting, 282–284
  relevance tuning, 254–255
  in SharePoint and Search Server 2008, 254
Ranking object, 310
RecordClick method, Web service query object model, 328
recovery options, index servers, 42–43
redundant array of inexpensive disks (RAID), 37
references
  adding to SharePoint DLLs, 320
  adding to VS projects, 289
registering custom security trimmers, 193, 198–200
registersecuritytrimmer command, stads tool, 193, 206
Registration method, Web service query object model, 328
regular expressions, Pattern triggers and, 165–166
relevance. See also ranking algorithms
anchor text and, 259–260
approaches to, 254–255
content management and, 255–258
custom dictionaries for, 268
defined, 253
diacritics in, 267–268
high-confidence matches and, 273–274
high-confidence results and, 276–279
keywords and best bets and, 274–276
language detection and, 260–262
metadata management and, 258–259
noise words in, 264–265
overview of, 253–254
ranking elements, 254
scopes and, 258
sorting people searches by, 225
thesaurus and, 265–267
URL depth and, 260
URL matching and, 279–280
user interface and, 270–273
user query behavior and, 269–270
word breakers and stemmers and, 262–264
replacement sets, 267
reports
overview of, 78–79
query, 78–79
results, 79
research services, Microsoft Office support, 329–330
restrictions
federated searches, 177
narrowing search results, 313
results
freshness of data and, 158–159
removal, 78, 258
reports, 79
server name mappings and, 77
unified results sets with federated searches, 158
results page, people searches, 147–149
Results per page property, Federated Search Results web parts, 186
results queries, 152
Results Query Options property, Federated Search Results web parts, 186
ResultTableCollection, 323
Retrieve Results Asynchronously property
Federated Search Results web parts, 186
Top Federated Results web part, 188
RevertToSelf impersonation, 201
root template, XSLT, 227
RunWithElevated Privileges, 189
S
Sample Data sections, federated searches, 176
Scalar element, in BDC metadata model, 106
scheduling searches, 63–64, 295–297
schema management, 304–308
changing property weighting, 307
crawled properties, 305
creating/mapping new managed property, 306–307
managed properties, 306
overview of, 304–305
query categories, 305
Schema object, 304–305
scopes
BDC data searches, 119–120
relevance tuning with, 258
search scopes, 75–77
working with, 302–304
Scopes class, 302–303
SCT (Search Center with Tabs). See also tabbed user interface
overview of, 145
results page, 147–149
search page and tabs for, 146–147
search applications
building queries, 312
descriptions, 308–312
element syntax, 308
example queries, 318–320
keyword syntax for, 313–314
overview of
query object models. See query object models
search applications (continued)

configured/administering. See configuring/administering searches
flow of typical, 215–218
logs, 311
results. See results
scheduling, 63–64, 295–297

searches, custom, 215
adding custom webparts, 251
adding new managed property, 241–243
adding/rendering custom properties, 237–241
Advanced Search Results web part, 244–245
autodiscovery of term definitions, 231
Core Search Results web part, 243–244
flow of typical searches and, 215–218
Hit Highlighting and, 235–237
overview of, 215
people searches and, 225
people searches and results using custom properties, 246–250
restarting Search service, 224
Search Center and, 218–219
Search Community Toolkit, 251–252
SPD support, 225, 231–235
stemming, 224
tabbed user interface, 219–222
thesaurus and synonyms, 222–224
XSLT for, 225–227
XSLT templates, 227–231

searches, Enterprise
algorithms and, 2–3
content types and, 1–2
deploying. See Enterprise Search deployment
products for. See Enterprise Search products
reasons for, 1
security, 2
Web protocols and, 3–4

security
architecture, 189–190
best bets and, 191
Enterprise Search and, 2
indexing and, 191

Search Box, in user interface, 270–271
Search Center
custom searches and, 218–219
people searches and, 145
tabbed user interface, 273
user accessing, 215
web parts in, 216–217
Search Center with Tabs. See SCT (Search Center with Tabs)
Search Community Toolkit, Codeplex, 251–252, 330
search engines, factors in user satisfaction, 253
Search Federation, for scaling searches, 43
Search Index on this Server, federating to local index, 171–172
Search Object model, 189
Search Options pane, people searches, 150
search pages
creating and editing, 221–222
people searches, 146–147
Search Results template, XSLT, 230
Search Server 2008. See Microsoft Search Server 2008
Search Service tool, Search Community Toolkit, 330–331
search services
managing, 52–57
overview of, 45–46
resetting, 82
restarting SharePoint search service, 224
SharePoint, 46–50
SSP (Shared Services Provider) and, 28
Timeout settings, 55
topologies, 29
WSS, 50–52
search.asmx file, 325
SearchContext object, 290–291
searches
algorithms, 2–3
backing up, 41–42
IRM-protected documents and, 191
overview of, 189
securing Search Server 2008, 212–213
security descriptors, 190
security timers, WSS search, 16
security trimming
  debugging custom trimmers, 200
  federated searches and, 180–181
  implementing custom trimmers, 191–193
  performance considerations, 193
  registering custom trimmers, 193, 198–200
  signing trimmer assembly (DLL) and adding to GAC, 198
  writing custom trimmer, 194–198
security trimming, BDC
  authentication, 201
  debugging custom trimmers, 209–212
  deploying custom trimmers, 206–209
  overview of, 201
  performance considerations, 202
  searches and, 202
  writing custom trimmers, 202–206
SELECT statements, SQL syntax in search applications, 316
server farms, SharePoint
  backing up, 41–42
  Index server recovery options, 42–43
  managing search services, 53–54
  medium farm environment, 31–32
  securing Search Server 2008, 212–213
  small farm environment, 30
  three-server farm environment, 31
server name mappings, 77
setup, Search Server 2008, 16–18
Shared Services Providers. See SSPs
  (Shared Services Providers)
SharePoint Designer. See SPD (SharePoint Designer)
SharePoint lists
  BDC in, 113–115
  cross-list capabilities, 331
SharePoint search service
  Microsoft Office support for, 329–330
  overview of, 46–50
  restarting, 224
SharePoint Server Search. See Microsoft Office SharePoint Server Search
SharePointContentSource, 293–294
Show More Results link, Federated Search Results web parts, 187
single server environment
  Search Server 2008 and, 212
  SharePoint search and, 29–30
single sign on (SSO), 201
small farm environment, 30
social distance, sorting people searches by, 225
social networking, colleagues and, 148–149
software boundaries, SharePoint search and, 33–34
sorting options, people searches, 225
SPD (SharePoint Designer)
  BDC and, 120–122
  customizing searches with, 225
  writing XSLT with, 231–235
SpecificFinder/Finder methods
  in BDC metadata model, 102–104
  working with BDC entities, 125–126
SPSecurity method, 189
SQL (Structured Query Language), 315–318
  data types, 315
  example queries, 318–320
  predicates, 316–318
  query syntax, 315–316, 318
SQL Server
  full-text search capabilities of, 25
  storing people information in SQL Server database, 131
SSO (single sign on), 201
SsoTicker filter, 99
SSPs (Shared Services Providers)
  administering SSP searches, 59–60
  associating security trimmers with, 193
  connecting BDC to Shared Services database, 125
  CPU requirements and, 35
  creating/editing SSP settings, 59
  GUID (Globally Unique Identifier) for, 222–223
  overview of, 58
  Search service and, 28
static ranking elements

static ranking elements, 254
Status method, Web service query object model, 328
stemming
custom searches and, 224
expansion sets and, 223
relevance and, 262–264
storage, hardware requirements for Enterprise searches, 37
string literals, SQL syntax in search applications, 315
Structured Data, connectors for, 155
Structured Query Language. See SQL (Structured Query Language)
stsadm tool
registering custom security trimmers, 198–200, 206
registersecuritytrimmer command, 193
search backups, 42
subdirectories, crawling, 293
synonyms
custom searches and, 222–224
expansion sets for, 266
working with, 309–310

T

tabbed user interface. See also SCT (Search Center with Tabs)
customizing searches, 219–222
Search Center, 273
search options, 218
templates, XSLT, 227–231
testing, Enterprise Search deployment, 38
thesaurus
custom searches and, 222–224
relevance and, 265–267
threads, performance monitoring settings, 40–41
three-server farm environment, 31
Timeout settings, search services, 55
titles
customizing title of federated results web part, 174–175
extraction algorithm for, 3, 282

Top Federated Results web part, 187–188
topology options, Enterprise Search deployment, 29
triggers, federated searches, 163–165
Always, 164
overview of, 163–164
Pattern, 164–165
Prefix, 164
troubleshooting BDC, 129
Trusted Subsystem, BDC authentication, 201
tsxxx.xml files, 223
TypeDescriptors, TypeDescriptor, DefaultValue elements, in BDC metadata model, 106–108

U

ULS (Unified Logging Service), 209–210
Unicode format, 268
Unified Logging Service (ULS), 209–210
URL depth
relevance and, 260
search algorithms, 3
URL matching
relevance and, 279–280
search algorithms, 3
URLs
authoritative and nonauthoritative pages and, 281–282
syntax in search applications, 314
Use Location Visualization property,
Federated Search Results web parts, 186
user credentials, federated searches, 178–179
user interface, 270–273
Advanced Search page, 272
Automatic Definitions, 273
Did you mean? feature, 273
duplicate collapsing, 271–272
highlighting with custom XSL, 271
interaction with searches, 215
Search Box in, 270–271
tabs, 273
user profiles
- BDC supplemental properties, 142–143
- configuring BDC import, 143–145
- configuring profile imports, 134–138
- configuring profile properties, 139–142
- managing, 133
- overview of, 131–133
- Profile Services connections, 133–134
- properties, 132, 138–139
- User Profile Properties page, 246

user query behavior, relevance and, 269–270
- UserContext filter, searching LOB systems, 99
- Username filter, searching LOB systems, 99
- UserProfile filter, searching LOB systems, 99

V
- ViewAccessor element, in BDC metadata model, 104
- VS (Visual Studio)
  - adding references to VS projects, 289
  - debugging custom trimmers, 200
  - writing custom trimmers, 194
  - writing XSLT with, 231

W
- WDS (Windows Desktop Search), 5–6
- web parts
  - adding custom, 251
  - BDC, 110–112
  - custom results and, 243–244
  - custom searches and, 218
  - inheriting from, 251
  - people searches and, 225
  - in Search Center, 216–217
- web parts, BDC
  - Business Data Actions web part, 112
  - Business Data Item Builder web part, 112
  - Business Data Item web part, 111–112
  - Business Data List web part, 110
  - Business Data Related List web part, 110–111
  - overview of, 109
- Web protocols, 3–4
- Web service query object model, 324–330
  - connecting to web service and creating proxy, 325
  - methods, 326–328
  - overview of, 324–325
  - performing query and getting results, 325–326
- web services
  - connecting to and creating proxy, 325
  - WSDL standard, 324–325
  - WSS search and, 15
- Web Services Description Language (WSDL), 324–325
- Web sites, connectors for, 155
- WebContentSource
  - creating, 294–295
  - overview of, 293–294
- WHERE clause, SQL syntax in search applications, 316
- wildcards
  - searching LOB systems, 100
  - stemming compared with wildcard searches, 224
- Windows Desktop Search (WDS), 5–6
- Windows Live, 24
- Windows Messenger, 225
- Windows SharePoint Services (WSS) Search. See WSS (Windows SharePoint Services) Search
- Windows Vista Search, 6–11
  - features in, 6
  - Instant Search, 9–11
  - searching Outlook 2007, 6–7
  - searching SharePoint from, 8–9
  - Word wheeling (search as you type), 6–7
- word breakers
  - custom dictionaries and, 268
  - overview of, 262–264
  - selection of, 261
- Word wheeling (search as you type), Windows Vista Search, 6–7
- WSDL (Web Services Description Language), 324–325
WSS (Windows SharePoint Services) Search

WSS (Windows SharePoint Services) Search, 11–16
  authentication, 180
  configuring search services, 50–52
  configuring searches, 14
  crawling content with, 12–13
  federated searches. See federated searches
  overview of, 11
  platform services, 14–16
  Search Server 2008 extending capabilities of, 20
  searching content with, 13–14

X

XML (Extensible Markup Language)
  BDC samples, 109
  case-sensitivity, 90

  returning results of federated searches as, 156
  XSLT for transforming, 215
XSD (Extensible Schema Definition), 87
XSL (Extensible Style Language)
  hit highlighting with, 271
  writing custom XSL, 175
XSLT (Extensible Style Language Transformation)
  customizing search results, 225–227
  root template, 227
  templates, 229–231
  transforming web part parameters, 228
  transforming XML using web parts, 215
Take your library wherever you go.

Now you can access more than 200 complete Wrox books online, wherever you happen to be! Every diagram, description, screen capture, and code sample is available with your subscription to the Wrox Reference Library. For answers when and where you need them, go to wrox.books24x7.com and subscribe today!