# CONTENTS AT A GLANCE

*Introduction*  

<table>
<thead>
<tr>
<th>PART 9  Creating High-Availability Applications</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 46  Understanding High Availability</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER 47  Monitoring System Performance</td>
<td>17</td>
</tr>
<tr>
<td>CHAPTER 48  Scaling with ColdFusion</td>
<td>43</td>
</tr>
<tr>
<td>CHAPTER 49  Scaling with J2EE</td>
<td>71</td>
</tr>
<tr>
<td>CHAPTER 50  Managing Session State in Clusters</td>
<td>93</td>
</tr>
<tr>
<td>CHAPTER 51  Deploying Applications</td>
<td>109</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 10  Ensuring Security</th>
<th>127</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 52  Understanding Security</td>
<td>129</td>
</tr>
<tr>
<td>CHAPTER 53  Securing the ColdFusion Administrator</td>
<td>153</td>
</tr>
<tr>
<td>CHAPTER 54  ColdFusion Security Options</td>
<td>161</td>
</tr>
<tr>
<td>CHAPTER 55  Creating Server Sandboxes</td>
<td>185</td>
</tr>
<tr>
<td>CHAPTER 56  Security in Shared and Hosted Environments</td>
<td>205</td>
</tr>
<tr>
<td>CHAPTER 57  Using the Administrator API</td>
<td>233</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART 11  Extending ColdFusion</th>
<th>249</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAPTER 58  Using Server-Side HTTP and FTP</td>
<td>251</td>
</tr>
<tr>
<td>CHAPTER 59  Creating and Consuming Web Services</td>
<td>277</td>
</tr>
<tr>
<td>CHAPTER 60  Working with Feeds</td>
<td>305</td>
</tr>
<tr>
<td>CHAPTER 61  Interacting with the Operating System</td>
<td>317</td>
</tr>
<tr>
<td>CHAPTER 62  Server-Side Printing</td>
<td>355</td>
</tr>
<tr>
<td>CHAPTER 63  Interacting with Directory Services</td>
<td>365</td>
</tr>
<tr>
<td>CHAPTER 64  Integrating with Microsoft Exchange</td>
<td>391</td>
</tr>
<tr>
<td>CHAPTER 65  Integrating with SharePoint and Portals</td>
<td>421</td>
</tr>
<tr>
<td>CHAPTER 66  Integrating with .NET</td>
<td>445</td>
</tr>
<tr>
<td>CHAPTER 67  Integrating with Microsoft Office</td>
<td>469</td>
</tr>
<tr>
<td>CHAPTER 68  Extending ColdFusion with Java</td>
<td>501</td>
</tr>
<tr>
<td>CHAPTER 69  Exposing ColdFusion Services</td>
<td>539</td>
</tr>
<tr>
<td>CHAPTER 70  Working with Gateways</td>
<td>557</td>
</tr>
<tr>
<td>CHAPTER 71  Integrating with SMS and IM</td>
<td>581</td>
</tr>
</tbody>
</table>

*Index*  

603
## CONTENTS

*Introduction*  

**PART 9  Creating High-Availability Applications**  

**CHAPTER 46 Understanding High Availability**  
High Availability Explained  
How Do I Know My Server Load?  
Load and Performance Testing  
High-Availability Plan: Seven Must-Haves for Building High-Availability Solutions  
Implement a Load-Balanced Web-Server Cluster  
Choose a Network Provider with Circuit Redundancy  
Install a Firewall  
Use RAID Level 5 on Database Servers  
Calculate Acceptable Risk  
Redundant Server Components vs. Commodity Computing  
Disaster Planning  
Some Truths About Web Hosting  
Active Site Monitoring  
The Quality-of-Service Guarantee  
What Next?  

**CHAPTER 47 Monitoring System Performance**  
Monitoring Your ColdFusion 9 Server  
Developing a Baseline  
Historical Analysis  
Analyzing ColdFusion Log Files  
JRun Metrics  
Analyzing Web Server Log Files  
Active Monitoring  
Server Probes  
Other Probes  
The ColdFusion Server Monitor  
The ColdFusion Server Manager  
Deciding What to Do Next  
Issues to Consider When Looking for Performance Problems  
Typical Bottlenecks  

**CHAPTER 48 Scaling with ColdFusion**  
The Importance of Scaling  
Scaling Considerations  
Tiered Application Architecture  
Server and Hardware Components  
Considerations for Choosing a Load-Balancing Option
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-Request Distribution Algorithms</td>
<td>51</td>
</tr>
<tr>
<td>Session State Management</td>
<td>51</td>
</tr>
<tr>
<td>Failover</td>
<td>51</td>
</tr>
<tr>
<td>Mixed Web Application Environments</td>
<td>52</td>
</tr>
<tr>
<td>How to Write ColdFusion 9 Applications That Scale</td>
<td>52</td>
</tr>
<tr>
<td>Code Organization</td>
<td>52</td>
</tr>
<tr>
<td>Modularity</td>
<td>53</td>
</tr>
<tr>
<td>Streamlined, Efficient Code</td>
<td>53</td>
</tr>
<tr>
<td>Avoiding Common Bottlenecks</td>
<td>54</td>
</tr>
<tr>
<td>Keeping Web Site Servers in Sync</td>
<td>57</td>
</tr>
<tr>
<td>What to Maintain?</td>
<td>57</td>
</tr>
<tr>
<td>Archive and Deploy</td>
<td>58</td>
</tr>
<tr>
<td>Other Options</td>
<td>58</td>
</tr>
<tr>
<td>Multiple ColdFusion 9 Instances</td>
<td>59</td>
</tr>
<tr>
<td>Hardware versus Software Load-Balancing Options</td>
<td>62</td>
</tr>
<tr>
<td>ColdFusion 9 Load Balancing and Software-Based Load Balancing</td>
<td>62</td>
</tr>
<tr>
<td>Dedicated Load-Balancing Hardware</td>
<td>63</td>
</tr>
<tr>
<td>Scaling with ColdFusion 9 Load Balancing</td>
<td>65</td>
</tr>
<tr>
<td>Understanding ColdFusion 9 Load Balancing</td>
<td>65</td>
</tr>
<tr>
<td>Configuring a Load Balanced Cluster</td>
<td>65</td>
</tr>
<tr>
<td>Hardware Load-Balancing Options</td>
<td>68</td>
</tr>
<tr>
<td>Load-Balancing Algorithms</td>
<td>69</td>
</tr>
<tr>
<td>BigIP</td>
<td>70</td>
</tr>
<tr>
<td>Finishing Up</td>
<td>70</td>
</tr>
<tr>
<td><strong>CHAPTER 49</strong> Scaling with J2EE</td>
<td>71</td>
</tr>
<tr>
<td>Benefits of Deploying ColdFusion on J2EE Application Server</td>
<td>72</td>
</tr>
<tr>
<td>Standards-Based</td>
<td>72</td>
</tr>
<tr>
<td>Multiple Platforms</td>
<td>72</td>
</tr>
<tr>
<td>Support for Legacy Infrastructure and Prior Investments</td>
<td>73</td>
</tr>
<tr>
<td>Inheritance of Multiplatform Interoperability</td>
<td>74</td>
</tr>
<tr>
<td>Development Values of ColdFusion 9 on J2EE</td>
<td>74</td>
</tr>
<tr>
<td>Deploying Rich Internet Applications in Java</td>
<td>74</td>
</tr>
<tr>
<td>Extending the Platform</td>
<td>75</td>
</tr>
<tr>
<td>Ease of Development</td>
<td>76</td>
</tr>
<tr>
<td>Leveraging Diverse Developer Skill Sets</td>
<td>76</td>
</tr>
<tr>
<td>Using J2EE Built-In Security Features</td>
<td>76</td>
</tr>
<tr>
<td>Improving Web Site Scalability</td>
<td>77</td>
</tr>
<tr>
<td>How ColdFusion Lives on Top of Java Server</td>
<td>79</td>
</tr>
<tr>
<td>Coding Implications from the Developer's Perspective</td>
<td>80</td>
</tr>
<tr>
<td>Scaling with ColdFusion and JRun</td>
<td>81</td>
</tr>
<tr>
<td>Scaling with Other Java Application Servers</td>
<td>85</td>
</tr>
<tr>
<td>Tuning the Java Virtual Machine</td>
<td>86</td>
</tr>
<tr>
<td>Introducing the JVM</td>
<td>86</td>
</tr>
<tr>
<td>Contents</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>Encryption 131</td>
<td></td>
</tr>
<tr>
<td>Cleartext Risks 131</td>
<td></td>
</tr>
<tr>
<td>ColdFusion Encryption Functions 131</td>
<td></td>
</tr>
<tr>
<td>How <code>Encrypt()</code> and <code>Decrypt()</code> Work 132</td>
<td></td>
</tr>
<tr>
<td><code>Encrypt()</code> and <code>Decrypt()</code> Parameters 136</td>
<td></td>
</tr>
<tr>
<td>ColdFusion’s <code>Hash()</code> Function 138</td>
<td></td>
</tr>
<tr>
<td>Browser Validation 143</td>
<td></td>
</tr>
<tr>
<td>Cross-Site Scripting (XSS) 144</td>
<td></td>
</tr>
<tr>
<td>Form and URL Hacks 146</td>
<td></td>
</tr>
<tr>
<td>Validation Techniques 147</td>
<td></td>
</tr>
<tr>
<td>File Uploads 148</td>
<td></td>
</tr>
<tr>
<td>Authentication and Authorization 150</td>
<td></td>
</tr>
<tr>
<td>CHAPTER 53 Securing the ColdFusion Administrator 153</td>
<td></td>
</tr>
<tr>
<td>Administrator Security Options 153</td>
<td></td>
</tr>
<tr>
<td>Administrator Page 153</td>
<td></td>
</tr>
<tr>
<td>RDS Page 155</td>
<td></td>
</tr>
<tr>
<td>User Manager Page 156</td>
<td></td>
</tr>
<tr>
<td>Creating a New User 157</td>
<td></td>
</tr>
<tr>
<td>Allowed IP Addresses 159</td>
<td></td>
</tr>
<tr>
<td>CHAPTER 54 ColdFusion Security Options 161</td>
<td></td>
</tr>
<tr>
<td>ColdFusion Security Framework 161</td>
<td></td>
</tr>
<tr>
<td>Development Security 162</td>
<td></td>
</tr>
<tr>
<td>User Security 163</td>
<td></td>
</tr>
<tr>
<td>Basic ColdFusion Login Example 170</td>
<td></td>
</tr>
<tr>
<td>ColdFusion Login Wizard Extension 173</td>
<td></td>
</tr>
<tr>
<td>ColdFusion Login Wizard Example 179</td>
<td></td>
</tr>
<tr>
<td>CHAPTER 55 Creating Server Sandboxes 185</td>
<td></td>
</tr>
<tr>
<td>Understanding Sandboxes 185</td>
<td></td>
</tr>
<tr>
<td>Understanding File and Directory Permissions 186</td>
<td></td>
</tr>
<tr>
<td>Changes in ColdFusion MX 187</td>
<td></td>
</tr>
<tr>
<td>Creating and Securing Applications Using Sandboxes 190</td>
<td></td>
</tr>
<tr>
<td>Enabling Sandbox Security 191</td>
<td></td>
</tr>
<tr>
<td>Adding a Sandbox 193</td>
<td></td>
</tr>
<tr>
<td>Configuring Your Sandbox 194</td>
<td></td>
</tr>
<tr>
<td>Sandbox Considerations 200</td>
<td></td>
</tr>
<tr>
<td>The Blackbox Sandbox Example 201</td>
<td></td>
</tr>
<tr>
<td>CHAPTER 56 Security in Shared and Hosted Environments 205</td>
<td></td>
</tr>
<tr>
<td>CFML-Based Risks 205</td>
<td></td>
</tr>
<tr>
<td>Securing the ColdFusion ServiceFactory 208</td>
<td></td>
</tr>
<tr>
<td>Securing RDS 208</td>
<td></td>
</tr>
<tr>
<td>Enabling RDS Password Security 209</td>
<td></td>
</tr>
<tr>
<td>Disabling RDS on Production Servers 209</td>
<td></td>
</tr>
<tr>
<td>Applying ColdFusion Sandbox Security 210</td>
<td></td>
</tr>
<tr>
<td>Contents</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td></td>
</tr>
<tr>
<td>Securing Multiple Server Configurations</td>
<td>211</td>
</tr>
<tr>
<td>Configuration Issues for Multi-homed Web Servers</td>
<td>212</td>
</tr>
<tr>
<td>Configuring ColdFusion for Multiple Hosts</td>
<td>214</td>
</tr>
<tr>
<td>Microsoft IIS</td>
<td>215</td>
</tr>
<tr>
<td>Creating the JRun Connector ISAPI Filter and the JRunScripts Virtual Directory</td>
<td>215</td>
</tr>
<tr>
<td>Adding the ColdFusion Extensions</td>
<td>216</td>
</tr>
<tr>
<td>Sun Java System Web Server</td>
<td>218</td>
</tr>
<tr>
<td>Other Issues</td>
<td>221</td>
</tr>
<tr>
<td>Running ColdFusion as a User</td>
<td>221</td>
</tr>
<tr>
<td>Adobe Flex and LiveCycle Integration</td>
<td>223</td>
</tr>
<tr>
<td>Disabling JSP Functionality</td>
<td>225</td>
</tr>
<tr>
<td>Securing the CFIDE Directory</td>
<td>225</td>
</tr>
<tr>
<td>Limiting Session Timeout Values</td>
<td>227</td>
</tr>
<tr>
<td>Removing Help Docs and Example Applications</td>
<td>227</td>
</tr>
<tr>
<td>Setting Debugging Restrictions</td>
<td>227</td>
</tr>
<tr>
<td>Encrypting ColdFusion Templates</td>
<td>229</td>
</tr>
<tr>
<td>Handling Error Messages</td>
<td>229</td>
</tr>
<tr>
<td>Setting Custom Tag Paths</td>
<td>230</td>
</tr>
<tr>
<td>Setting the File Name Extension</td>
<td>231</td>
</tr>
<tr>
<td>Adding the Default Document</td>
<td>232</td>
</tr>
<tr>
<td>Staying Informed</td>
<td>232</td>
</tr>
<tr>
<td>Using the Administrator API</td>
<td>233</td>
</tr>
<tr>
<td>Understanding the Admin API</td>
<td>233</td>
</tr>
<tr>
<td>Building Custom Admin Consoles</td>
<td>235</td>
</tr>
<tr>
<td>The Façade Component</td>
<td>235</td>
</tr>
<tr>
<td>Security Implications for the Admin API</td>
<td>242</td>
</tr>
<tr>
<td>ColdFusion ServiceFactory</td>
<td>242</td>
</tr>
<tr>
<td>Admin API Security Risks</td>
<td>242</td>
</tr>
<tr>
<td>Securing the Admin API</td>
<td>243</td>
</tr>
<tr>
<td>Securing the Façade Component</td>
<td>244</td>
</tr>
<tr>
<td>Admin API Best Practices</td>
<td>247</td>
</tr>
<tr>
<td>Admin API Configuration</td>
<td>248</td>
</tr>
<tr>
<td>Custom Console Coding Best Practices</td>
<td>248</td>
</tr>
<tr>
<td>PART II Extending ColdFusion</td>
<td>249</td>
</tr>
<tr>
<td>CHAPTER 58 Using Server-Side HTTP and FTP</td>
<td>251</td>
</tr>
<tr>
<td>Overview</td>
<td>251</td>
</tr>
<tr>
<td>Using &lt;cfhttp&gt;</td>
<td>252</td>
</tr>
<tr>
<td>Errors and Results for a &lt;cfhttp&gt; Call</td>
<td>255</td>
</tr>
<tr>
<td>Using the &lt;cfhttpparam&gt; Tag</td>
<td>256</td>
</tr>
<tr>
<td>Putting &lt;cfhttp&gt; to Use</td>
<td>257</td>
</tr>
<tr>
<td>Using the GET Method</td>
<td>257</td>
</tr>
<tr>
<td>Building a Query from a Text File</td>
<td>259</td>
</tr>
<tr>
<td>Using HTTP with Scripting</td>
<td>261</td>
</tr>
</tbody>
</table>
Contents

Using the POST Method 262
Summarizing the <cfhttp> Tag 264
Using <cfftp> 264
  Connection Operations with <cfftp> 264
  File and Directory Operations with <cfftp> 267
  Errors and Results for a <cfftp> Call 269
Putting <cfftp> to Use 271
  Displaying Available Files 271
  Using <cfftp> to Download a File 273
  Using <cfftp> to Upload a File 274
Secure FTP Connections 275
Using FTP with Scripting 275
Summarizing the <cfftp> Tag 276

CHAPTER 59 Creating and Consuming Web Services 277
Understanding Web Services 277
  Business Models 278
  Core Technologies 278
WSDL 279
Creating Web Services 286
  Components 287
  Defining Complex Data Types 291
Consuming Web Services 295
  Consult the WSDL 295
  Invoking ColdFusion Web Services 296
  Complex Data Types 297
  Invocation with ColdFusion Builder 298
  Invoking .NET Web Services 299
  Dealing with Generated WSDL 299
Working with SOAP Requests 300
  Calling Web Services with Nillable Arguments 301
Best Practices 302
  Error Handling 303
  Configuring Web Services in the ColdFusion Administrator 303

CHAPTER 60 Working with Feeds 305
Why Use Feeds? 305
Flavors of RSS and Atom 306
Creating Feeds 306
Reading Feeds 310
  Using Dublin Core and iTunes Extensions 314

CHAPTER 61 Interacting with the Operating System 317
Introduction to <cffile> 317
Accessing the Server’s File System 319
  Reading and Writing Files 319
  Copying, Moving, Renaming, and Deleting Files 321
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uploading Files</td>
<td></td>
</tr>
<tr>
<td>What If the File Already Exists on the Server?</td>
<td>324</td>
</tr>
<tr>
<td>Determining the Status of a File Upload</td>
<td>325</td>
</tr>
<tr>
<td>Building an Upload Interface</td>
<td>326</td>
</tr>
<tr>
<td>Handling Multiple Uploads at Once</td>
<td>331</td>
</tr>
<tr>
<td>Using File Functions</td>
<td></td>
</tr>
<tr>
<td>Getting File Information</td>
<td>333</td>
</tr>
<tr>
<td>Reading Files with <code>&lt;cfloop&gt;</code></td>
<td>334</td>
</tr>
<tr>
<td>Manipulating Folders on the Server with <code>&lt;cfdirectory&gt;</code></td>
<td>335</td>
</tr>
<tr>
<td>Using <code>&lt;cfdirectory&gt;</code></td>
<td>335</td>
</tr>
<tr>
<td>Getting the Contents of a Directory</td>
<td>337</td>
</tr>
<tr>
<td>Building a Simple File Explorer</td>
<td>337</td>
</tr>
<tr>
<td>Working with Virtual Files</td>
<td></td>
</tr>
<tr>
<td>Executing Programs on the Server with <code>&lt;cfexecute&gt;</code></td>
<td>345</td>
</tr>
<tr>
<td>Interacting with the System Registry Using <code>&lt;cfregistry&gt;</code></td>
<td>347</td>
</tr>
<tr>
<td>Working with Zip Files</td>
<td></td>
</tr>
<tr>
<td>Creating Zip Files</td>
<td>348</td>
</tr>
<tr>
<td>Expanding Zip Files</td>
<td>349</td>
</tr>
<tr>
<td>Listing, Reading, and Deleting from Zip Files</td>
<td>350</td>
</tr>
<tr>
<td>Working with <code>&lt;cfzipparam&gt;</code></td>
<td>352</td>
</tr>
<tr>
<td><strong>CHAPTER 62 Server-Side Printing</strong></td>
<td></td>
</tr>
<tr>
<td>Overview</td>
<td>355</td>
</tr>
<tr>
<td>Using <code>&lt;CFPRINT&gt;</code></td>
<td></td>
</tr>
<tr>
<td>Printing to the Default Printer</td>
<td>357</td>
</tr>
<tr>
<td>Printing to Another Printer</td>
<td>357</td>
</tr>
<tr>
<td>Controlling Printer Capabilities</td>
<td></td>
</tr>
<tr>
<td>Primary <code>&lt;CFPRINT&gt;</code> Attributes</td>
<td>358</td>
</tr>
<tr>
<td>Printer-Dependent Attributes</td>
<td>358</td>
</tr>
<tr>
<td>The <code>fidelity</code> Attribute</td>
<td>360</td>
</tr>
<tr>
<td>Java Print Services and Attributes</td>
<td>360</td>
</tr>
<tr>
<td>Additional <code>&lt;CFPRINT&gt;</code> Attributes</td>
<td>360</td>
</tr>
<tr>
<td>Determining Available Printers</td>
<td>361</td>
</tr>
<tr>
<td>Determining Available Printers</td>
<td>361</td>
</tr>
<tr>
<td>Printer Permissions</td>
<td>362</td>
</tr>
<tr>
<td>Related Features</td>
<td></td>
</tr>
<tr>
<td>Print Log</td>
<td>362</td>
</tr>
<tr>
<td>Using <code>&lt;CFTTHREAD&gt;</code></td>
<td>362</td>
</tr>
<tr>
<td>Printing Other than PDFs</td>
<td>363</td>
</tr>
<tr>
<td><strong>CHAPTER 63 Interacting with Directory Services</strong></td>
<td></td>
</tr>
<tr>
<td>Understanding LDAP</td>
<td>365</td>
</tr>
<tr>
<td>Directory Structures</td>
<td>366</td>
</tr>
<tr>
<td>Name Conventions</td>
<td>367</td>
</tr>
<tr>
<td>Using <code>&lt;CFLDAP&gt;</code></td>
<td></td>
</tr>
<tr>
<td>Tag Overview</td>
<td>368</td>
</tr>
</tbody>
</table>
The ACTION Attribute 370
The SCOPE Attribute 372
The MODIFYTYPE Attribute 372
The SECURE Attribute 373
Interacting with Directories 373
Active Directory 373
Querying Directories 374
Adding Entries 377
Modifying Entries 379
Deleting Entries 385
Active Directory 386
Building an LDAP CFC 387

CHAPTER 64 Integrating with Microsoft Exchange 391
ColdFusion Exchange Server Tags 391
<cfexchangeconnection> 392
<cfexchangecalendar> 395
<cfexchangecontact> 400
<cfexchangefilter> 403
<cfexchangemail> 404
<cfexchangetask> 408
Managing Exchange Server Connections 411
Connection Requirements 411
Persistent Connections 412
Transient Connections 413
Delegate Account Access 414
Managing Exchange Server Items 414
Retrieving Exchange Items and Attachments 414
Deleting Exchange Items 416
Modifying Exchange Items 417

CHAPTER 65 Integrating with SharePoint and Portals 421
Web Portals 421
ColdFusion 9 Support for Integration with SharePoint 422
Accessing ColdFusion 9 from SharePoint 423
Using the SharePoint Page Viewer Web Part to Access a ColdFusion 9 Application 423
Using the CFSSOWebPart Web Part to Access a ColdFusion 9 Application 426
Other Approaches for Accessing ColdFusion Data and Content from SharePoint 432
Accessing SharePoint from ColdFusion 9 433
Using ColdFusion 9 to Consume SharePoint Web Services 433
Other Approaches for Accessing SharePoint Data and Content in ColdFusion 9 Applications 438
ColdFusion 9 Support for Integrating with Portal Servers 438
ColdFusion 9 Portlets 439
Creating ColdFusion Portlet CFCs 439
Creating Your First IM Application 586
Generating IM Messages 586
Making New Friends 587
User Status Management 587
The IM Gateway Helper 588
Creating Interactive Applications 592
Understanding SMS 595
Testing Your SMS Applications 596
Getting Your Messages Out 596
Defining SMS Gateways 597
Generating SMS Messages 599
Responding to SMS Messages 600
Ensuring Delivery of Your Messages 600
Extending Your Application Even Further 601

Index 603
Introduction

Who Should Use This Book?

This book is written for anyone who wants to create cutting-edge Web-based applications.

If you are a Webmaster or Web page designer and want to create dynamic, data-driven Web pages, this book is for you. If you are an experienced database administrator who wants to take advantage of the Web to publish or collect data, this book is for you, too. If you are starting out creating your Web presence, but know you want to serve more than just static information, this book will help get you there. If you have used ColdFusion before and want to learn what’s new in ColdFusion 9, this book is also for you. Even if you are an experienced ColdFusion user, this book provides you with invaluable tips and tricks and also serves as the definitive ColdFusion developer’s reference.

This book teaches you how to create real-world applications that solve real-world problems. Along the way, you acquire all the skills you need to design, implement, test, and roll out world-class applications.

How to Use This Book

This is the ninth edition of Adobe ColdFusion Web Application Construction Kit, and what started as a single volume a decade ago has had to grow to three volumes to adequately cover ColdFusion 9. The books are organized as follows:


- **Volume 2**—*Adobe ColdFusion 9 Web Application Construction Kit, Volume 2: Application Development* (ISBN 0-321-67919-9) contains Chapters 22 through 45 and covers the ColdFusion features and language elements that are used by most ColdFusion developers most of the time. (Chapters 43, 44, and 45 are online.)

- **Volume 3**—*Adobe ColdFusion 9 Web Application Construction Kit, Volume 3: Advanced Application Development* (ISBN 0-321-67920-2) contains Chapters 46 through 71 and covers the more advanced ColdFusion functionality, including extensibility features, as well as security and management features that will be of interest primarily to those responsible for larger and more critical applications.
These books are designed to serve two different, but complementary, purposes.

First, as the books used by most ColdFusion developers, they are a complete tutorial covering everything you need to know to harness ColdFusion’s power. As such, the books are divided into parts, or sections, and each section introduces new topics building on what has been discussed in prior sections. Ideally, you will work through these sections in order, starting with ColdFusion basics and then moving on to advanced topics. This is especially true for the first two books.

Second, the books are invaluable desktop references. The appendixes and accompanying Web site contain reference chapters that will be of use to you while developing ColdFusion applications. Those reference chapters are cross-referenced to the appropriate tutorial sections, so that step-by-step information is always readily available to you.

The following describes the contents of Adobe ColdFusion 9 Web Application Construction Kit, Volume 3: Advanced Application Development.

**Part IX: Creating High-Availability Applications**

Chapter 46, “Understanding High Availability,” introduces the basics of high availability, including load balancing, failover, quality of service (QoS), and clusters.

To address scalability and high availability, it is important to understand how to measure and gauge system performance. Chapter 47, “Monitoring System Performance,” introduces the monitoring tools provided by the underlying operating system as well as the powerful ColdFusion Server Monitor and the new Server Manager tool.

Chapter 48, “Scaling with ColdFusion,” analyzes and compares the various hardware- and software-based scalability solutions available to you, emphasizing the differences between them and any special issues that need to be addressed as a result.

Chapter 49, “Scaling with J2EE,” explores Java 2 Enterprise Edition–based scalability, as well as the benefits of running ColdFusion on top of this powerful platform.

Because session-state information is usually very server specific, creation of server clusters (or server farms) requires you to rethink how you manage session information. Chapter 50, “Managing Session State in Clusters,” teaches you how to manage sessions and session state across clusters when necessary and how to use J2EE-based session-state management.

Chapter 51, “Deploying Applications,” rounds out Part IX by teaching application deployment techniques and strategies.

**Part X: Ensuring Security**

Chapter 52, “Understanding Security,” explains security risks and introduces important security fundamentals, such as encryption, authentication, authorization, and access control.

ColdFusion is managed using the ColdFusion Administrator, a Web application written in ColdFusion itself. This application must be carefully secured, and Chapter 53, “Securing the ColdFusion Administrator,” discusses strategies to secure the application while still ensuring access to those who need it.
Chapter 54, “ColdFusion Security Options,” introduces ColdFusion’s security framework and explains how (and why) to use the underlying operating system’s security features.

Sandboxes allow the creation of virtual security entities to secure files, directories, data sources, and even CFML elements. Chapter 55, “Creating Server Sandboxes,” explains in detail how to use the powerful sandbox feature.

Chapter 56, “Security in Shared and Hosted Environments,” tackles the security concerns unique to shared and hosted servers. Server sandboxes are also explained, along with databases, remote access, and other important issues.

ColdFusion features an Administrator API, which can be used to build custom ColdFusion Administrative consoles and applications. Chapter 57, “Using the Administrator API,” introduces this powerful management tool.

Part XI: Extending ColdFusion

Chapter 58, “Using Server-Side HTTP and FTP,” teaches you how to use these Internet protocols from within your own code. With the help of these protocols, you can easily write applications that interact with other servers and services anywhere on the public Internet and private intranets, and even implement syndication services of your own.

ColdFusion can both create and consume Web services, providing integration with all sorts of systems and services. Chapter 59, “Creating and Consuming Web Services,” explains what Web services are and why they are of so much interest.

Another popular way to share data is via RSS and ATOM data feeds. ColdFusion supports both, allowing easy feed creation and consumption. Chapter 60, “Working with Feeds,” explains how to use this functionality in your own applications.

Chapter 61, “Interacting with the Operating System,” introduces the powerful and flexible ColdFusion `<cffile>` and `<cfdirectory>` tags. You learn how to create, read, write, and append local files; manipulate directories; and add file uploading features to your forms. You also learn how to spawn external applications when necessary, and how to use the virtual file system.

Chapter 62, “Server-Side Printing,” introduces the `<cfprint>` tag and explains how you can use it to generate printed output on the ColdFusion server.

Chapter 63, “Interacting with Directory Services,” covers directory services and Lightweight Directory Access Protocol (LDAP) and how to use both of them simply and easily via the `<cfldap>` tag.

Microsoft Exchange has become a critical tool for many organizations. ColdFusion features sophisticated Exchange integration, which can be used to provide calendaring, scheduling, and more within your applications. Chapter 64, “Integrating with Microsoft Exchange,” explains how to use the ColdFusion Exchange tags to access the power of Exchange programmatically.

Chapter 65, “Integrating with SharePoint and Portals,” continues the coverage of integration with Microsoft networking and productivity software with information about how to create portlets with ColdFusion, how to integrate with SharePoint single sign-on, and more.
Chapter 66, “Integrating with .NET,” explains .NET basics and teaches you how to interact with .NET objects and assemblies.

Chapter 67, “Integrating with Microsoft Office,” continues this discussion with detailed coverage of integration with Microsoft Office, with special emphasis on spreadsheet file integration.

ColdFusion is built on underlying Java infrastructure. Chapter 68, “Extending ColdFusion with Java,” teaches you how to combine the strengths of ColdFusion and its Java foundations to leverage the best of both worlds. Included is coverage of servlets, Enterprise JavaBeans (EJBs), and more.

ColdFusion exposes some core functionality via services that can be consumed by other servers and platforms. Chapter 69, “Exposing ColdFusion Services,” introduces this new services infrastructure and explains how to use it within your own applications.

Although primarily used to power Web applications, ColdFusion can interact with all sorts of systems and services via gateways. The ColdFusion gateway engine provides access to sockets, JMS, asynchronous processing, and more. Chapter 70, “Working with Gateways,” introduces gateway technology and explains how to use gateways as well as how to create your own.

Chapter 71, “Integrating with SMS and IM,” continues this topic with coverage of three specific gateways: the Short Message Service (SMS) gateway used to interact with SMS on devices, the Lotus Sametime gateway used to interact with that instant messaging (IM) technology, and the Extensible Messaging and Presence Protocol (XMPP) gateway used to interact with IM via XMPP.

**The Web Site**

The book’s accompanying Web site contains everything you need to start writing ColdFusion applications, including:

- Links to obtain ColdFusion 9
- Links to obtain Adobe ColdFusion Builder
- Source code and databases for all the examples in this book
- Electronic versions of some chapters
- An errata sheet, should one be required
- An online discussion forum


And with that, turn the page and start reading. In no time, you’ll be creating powerful applications powered by ColdFusion 9.
This page intentionally left blank
If you are reading this book, chances are your goal is not only to build a rock-solid ColdFusion application, but also to keep that application running at full speed through active and less-than-active times. At the beginning of the Internet boom, circa 1996, the Internet consisted of hundreds of pages of information, mostly published by universities and private individuals. Although these informational Web sites were important, if one of them was down for maintenance in the middle of the day, or if a Web server was overutilized on a Friday morning, nobody lost real business because few people were doing business on the Internet.

Those days are over. Businesses, organizations, and even governments are relying more and more on Internet-related revenue-generating activities such as selling products and communicating with business partners. Consequently, CIOs and CTOs alike are demanding better performance and more reliability from their Web sites. They now expect e-commerce sites to be profitable, making it more important than ever to maintain highly available Web sites. In today’s terms, downtime means thousands of dollars of lost revenue, and in some cases security and safety risks.

With the advent of broadband Internet connections and faster personal computers, consumers demand more and more from the Web sites they visit. If response times do not meet customer expectations, companies run the risk of damaging their public images. Reliance on the Internet as a tool to conduct business is increasing every day, and so is our ability to create scalable, stable environments for hosting Web sites.

Enter the concept of high availability. Because today’s Web applications must be available all the time without exception, and because today’s servers—though highly advanced—are still mechanical devices, you must put thought and planning into a Web application’s design to ensure its success. Fortunately, once you have the key pieces in place, a highly available Web application is often easier to manage than a standard Internet site.

The first few chapters of this volume show how to build a highly available ColdFusion site architecture, understand Web site performance, and allow the site to expand into the future. ColdFusion 9
is now more scalable, faster, and more robust than ever and supports architecture based on the latest Java technology standards (Java 1.6). This chapter gives you an idea of how to ascertain your current level of availability from within ColdFusion, and makes suggestions for understanding and improving your Web site’s uptime and strengthening its architecture.

**High Availability Explained**

*High availability* refers to your Web application’s capability to respond 99.99 percent of the time. You’ll achieve this figure, which works out to about one hour of downtime per year, by designing network architectures and Web applications that eliminate all single points of failure or that have a high degree of fault tolerance (redundancy at every level within the hosting provider, network, server, and Web-application architecture).

Here’s an example: You have a basic Web site that contains a single Web server and a single database server. One day a power surge causes a power supply failure in the Web server, and the site goes down. If that server’s running an e-commerce site, you might lose business irreparably. However, if you’ve built the site on a cluster of two or more Web servers, the end user can navigate the site normally and may never know that any component failure occurred. Ideally, all your servers would remain healthy all the time; however, that uptime percentage I mentioned earlier does not mean each server will maintain individual uptimes of 99.99 percent. Rather, this percentage refers to the Web application’s total uptime as seen by the end user. See Table 46.1, which describes uptime percentage and downtime per year for an application running continuously 24 hours a day, 7 days a week, and 365 days a year.

<table>
<thead>
<tr>
<th>UPTIME PERCENTAGE</th>
<th>DOWNTIME PER YEAR ALLOWED</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.999</td>
<td>Approximately 5 minutes</td>
</tr>
<tr>
<td>99.99</td>
<td>53 minutes</td>
</tr>
<tr>
<td>99.9</td>
<td>8 hours, 45 minutes</td>
</tr>
<tr>
<td>99</td>
<td>87 hours, 36 minutes</td>
</tr>
</tbody>
</table>

In the rest of this chapter, I’ll give you a conceptual idea of how to consider high availability when you are planning an application.


The largest problem many application developers and network engineers face is knowing precisely when a problem exists. To improve your Web site’s uptime and stability, first you must think about how to determine the site’s actual availability from a performance perspective. Most sites crash because of too great a load on the server and improper performance tuning.
How Do I Know My Server Load?

The amount of traffic on a Web server at any given time is called the load. The percentage load is a measure of that Web server’s utilization.

Load and Performance Testing

So, you are ready to launch your Web site. Before launching any Web application that you anticipate will generate moderate to large amounts of traffic, you should perform a structured server-load test. This is basically a calculated simulation of anticipated site traffic during a given period. The load test will assess the optimal performance of your Web site and help you define the maximum load it can handle. Ascertaining the maximum load a Web site or service will handle before crashing is called stress testing.

Using a performance-testing package, you can author scripts that generate a given number of requests during a given period (say, 8 hours) or simulate a given number of users or sessions. The performance-testing package generates a load on the server by simulating the click stream of multiple users and then reports the server response times. By gradually increasing the number of users you’re simulating and monitoring the server response times, you can project how much traffic will cause your Web server to go down. You can also model complex behavior such as peak times, sudden traffic spikes, and special conditions (such as users leaving), allowing you to create very accurate models of real-world system use.

Selecting a load-testing product can sometimes be difficult since there are a wide number of available tools. These tools range from open source free tools to tools costing tens of thousands of dollars. Some questions to keep in mind as you select a tool are how often will you use the tool, how complex are the tests that you need to run, how important is it that you get accurate performance results, whether you need to accurately simulate network connections, and whether you need to benchmark other parts of your application besides the application layer (such as third-party messaging tools or the database). Use these questions to help narrow your selection; then try a few products before selecting the one that’s right for you. Several packages the author has used include the following:

- WebLoad has an excellent free open source edition that is the best open source testing tool around (http://www.webload.org/). It also has a commercial version and plug-ins that support things like Flex and AMF test automation.
- Hewlett-Packard offers several options, including hosted load testing and software such as LoadRunner (http://www.hp.com).
- Keynote provides hosted, Web-based testing services (http://www.keynote.com).
- Empirix has a suite of products, including Hammer Test Engine (http://www.empirix.com).
- Open STA also offers a free open source testing tool (http://www.opensta.org).
Microsoft offers a free Web application stress tool (http://www.microsoft.com).

Searching the Web with Yahoo or Google found several sites discussing load testing, including Knowledge Storm, http://www.knowledgestorm.com, which listed many solutions and information on this subject. Typically, the more expensive solutions provide more functionality and can simulate more simultaneous users.

Here are some tips for preparing to load-test your Web site. First, compile site-usage statistics using your Web server’s statistics logs. If your site is new, attempt to estimate usage of your Web site. Estimating these statistics can be difficult. At the very least, try to estimate the peak number of users and/or sessions per hour and the most popular route through your site.

TIP

If you are developing a new application or site for which you want to do realistic testing, ask around on forums or email lists for people to share their applications statistics with you. Often organizations with similar sites will be happy to share at least some general statistics that will help you get an idea of what numbers to use in your testing.

These are some of the most important usage statistics for your Web site:

- Average number of users and/or sessions per hour
- Peak number of users and/or sessions per hour
- Most popular path through site based on analyzed traffic or real-use cases
- Most CPU-intensive Web pages or activities (such as logging in to the Web site, or performing database-intensive activity such as running queries and inputting large amounts of information)
- Most requested page(s) and top entry page(s)
- Average length of stay on site
- Most popular connection speeds used by visitors (56 Kbps, DSL or cable, T1, and so forth)
- Average response time or latency for pages
- CPU usage and other performance-monitoring statistics

After gathering your statistics or estimates, prepare test scripts and parameters. Test scripts simulate traffic patterns and usage throughout the site, and parameters set expectations for site performance.

A typical test script might include an area where users log in to the site and post information. The test script would simulate how users browse, log in, and post information on the site. For an e-commerce site, the test script might simulate users browsing for products, adding items to a shopping cart, and checking out.
How Do I Know My Server Load?

The site’s login sequence, shopping cart, and user checkout all query the database server. Including these sections of the site in the performance test is essential to ascertaining the Web server’s response time when making requests to the database server.

In general you want to make sure you cover these parameters in your test scripts:

- Maximum number of users and/or sessions to simulate (if your Web site’s peak number of users is, say, 500 per hour, you may want to test it for 1,000 users per hour to ensure that your site will not crash during peak usage)
- Length of sessions (each user stays on your site for an average of 5 minutes)
- Length of the test (usually a minimum 1-hour test with at least a 20- to 30-minute “smoke” test before you start your real testing)
- Ramp-up times (adding users and/or sessions gradually and sporadically to simulate real Web traffic)
- Connection speed mix (majority of test users will access the site over a 56 Kbps connection; others will access over DSL or cable connections)

Now it is time to prepare your Web site for the load test. First, deploy a good copy of your application to your testing server, or to the production server if the site is not live. It is best to use a server that exactly reflects your production environment thus accurately reflecting your live Web site’s performance. Second, turn on performance-monitoring tools. Third, perform the load test.

Assessing the results of the load test will provide valuable information pertaining to the Web site’s performance and bottlenecks. Most load-testing software provides statistics on users and/or sessions attempted per hour, concurrent users and/or sessions per minute, page latency or response time per hour, and errors encountered. The concurrent users and session statistics will indicate your Web site’s peak performance capability.

NOTE
Users do not always browse your site the way you want them to, so you may need to develop your test scripts to reflect this. One way to do this is simply to record real users or to once again mine your logs for information about what paths users follow and the average amount of time users spend on specific pages or actions.

TIP
Never load-test your site on your production servers if your application is live. You don’t want to crash your own Web site!

NOTE
Often called response time, latency is the delay experienced between the moment when a request is made to the server and the point at which the user can view the page.

If you run your performance test and notice that you have immediate problems with site response under very little simulated traffic, you have a bottleneck that requires examination. Typical bottlenecks for Web servers include CPU, memory, network, other servers (such as the database server), and code. Identifying and correcting bottlenecks before launching the site will help to avoid frustration and extra expense after launch.
Chapter 47 includes more detail on how to monitor and understand the performance of your Web servers, identify bottlenecks, and tune servers to run efficiently. Inability to handle the load is one of the most common causes for site failure, so knowing what to expect beforehand will put you ahead of the game.

NOTE
When configuring your Web and database servers, pay specific attention to any extra, nonessential software you load on each server. Even software as simple as an enterprise-monitoring agent or an antivirus program can have an impact on your server’s performance.

**High-Availability Plan: Seven Must-Haves for Building High-Availability Solutions**

You have seen all the monitoring reports, and you have responded to the ColdFusion alarms. You now have the information you need to start building a plan. Start by looking at the failure points.

Once you have a good idea of how much traffic your servers can take, it’s time to start building a plan to solidify the availability of your site and achieve that 99.99 percentile. The following action items are the most important considerations to ensure that your site will be up, available, and free of single points of failure that can dead-end site traffic:

- Implement a load-balanced Web server cluster to make server downtime invisible.
- Choose a network host that offers circuit redundancy.
- Install a correctly configured firewall to protect against unwanted visitors.
- Use RAID Level 5 on database servers.
- Implement a backup and recovery strategy and process.
- Calculate a level of risk that is both business-smart and cost-effective.
- Choose fault tolerance systems to reduce failure points.

The following sections describe each of these items in detail.

**Implement a Load-Balanced Web-Server Cluster**

The easiest and most effective way to make server downtime invisible and increase the availability of any site is to provide load balancing and failover for a Web server cluster. Use of load balancing devices allows the system to distribute traffic load evenly among all systems in your cluster, ensuring that no single server becomes unavailable due to intense load. Failover specifically applies when a server in your cluster becomes unresponsive due to a disaster such as software or hardware failure. Having a failover system allows your cluster to switch to backup hardware, seamlessly shifting traffic—for example, from the main database server to a backup database server.
Load balancing and failover accomplish two goals:

- Maximize server efficiency by balancing Web traffic between servers
- Redirect traffic from nonresponsive Web servers, allowing server failures to go unnoticed by the end user (this is the failover)

Load balancing technology comes in three flavors:

- Software-based
- Hardware-based
- Combination software and hardware

**Software-Based Load Balancing**

Adobe’s ColdFusion 9 Enterprise server includes the capability to cluster multiple instances of ColdFusion (described in Chapter 48). This capability allows you to use ColdFusion clustering for failover or as a software-based load balancer. Software-based load balancers communicate on the network level and maintain a heartbeat with other servers in the cluster to identify server health. If a server in the cluster fails to respond to the heartbeat, the server fails over—that is, traffic is redirected away from the affected server.

A number of open source and free open source software load balancing solutions are available, especially for Linux (http://1cic.org/load_balancing.html). However, software-based load balancing is usually only good for smaller systems, because at some point the software used to load-balance a cluster may begin to affect the cluster’s performance. This occurs because each machine has to spend some of its available resources running the clustering software, as well as sending and receiving information over the wire to determine which machines are running and busy, so that the software can decide where to route traffic. Hardware-based solutions are usually faster and much more reliable, and offer a number of features not included in software solutions.

**NOTE**

Server heartbeat is defined as continual communication of a server’s status to all other servers within the cluster and/or the load balancing software or device.

**Hardware-Based Load Balancing**

Cisco’s LocalDirector and F5’s BigIP series use a server-based architecture to load-balance in front of the Web server cluster. Each server-based load balancer works differently. Hardware-based load balancers are more efficient (and more costly) than software-based ones because they actively monitor each connection to each server in the cluster (rather than relying on the servers to manage their own connections and balance the load). The hardware load balancer contains the virtual address of the site (usually the www.domain.com name) and redirects traffic to each of the servers in the cluster according to a predefined algorithm (such as round robin or least connections). When the load balancer determines that a server is nonresponsive or is displaying bad content, the load balancer removes that server from the cluster.
Hardware load balancers are a better choice for high-traffic sites because they offload the cluster-management overhead onto a dedicated machine. In addition, they are more flexible when it comes to things like managing persistent (sticky) sessions and filtering traffic. It is generally best practice with any load balancing system (hardware or software) to make sure there is some redundancy. By configuring two hardware load balancers in tandem, you can set one to fail over in case the other goes down, thus eliminating the single point of failure inherent in placing a single server in front of your Web cluster. Figure 46.1 demonstrates how a hardware load balancer handles site traffic.

Figure 46.1
Typical hardware load-balancing configuration.

NOTE
Hardware load balancers are in general so cheap in relationship to what they offer that it is almost always better to use a hardware load balancer rather than a software load balancer, especially if you are using more than two machines.

Combination Software and Hardware Load Balancing
Using ColdFusion 9’s clustering in tandem with a hardware load balancer, you can combine the monitoring and reporting capabilities of ColdFusion 9 with the cluster-management features of a hardware load balancer. ColdFusion can also supply redundancy if the hardware load balancer fails.

Choose a Network Provider with Circuit Redundancy
When most users type a Web address into their browser, they do not realize that data can go through 10 to 15 stops en route to the destination Web server. These stops (called hops) can be local routers, switches, or large peering points where multiple network circuits meet. The Internet
really is similar to a superhighway, and like any congested highway, it’s prone to traffic jams (called *latency*). As far as your users are concerned, your site is down if there are any problems along the route to your site, even if your ColdFusion servers are still alive and ready to deliver content. Imagine that you are driving along the freeway on a Monday morning and it becomes congested. Knowing an alternate route will allow you to move around the congestion and resume your prior course. Hosting your Web applications on a redundant network allows them to skirt traffic problems in a similar fashion.

Always choose a hosting provider that can implement redundant network circuits (preferably two major Tier 1 upstream providers such as AT&T, Global Crossing, Level 3, or Sprint). Many hosting providers have multiple circuits from multiple providers configured with Border Gateway Protocol (BGP). A BGP configuration enables edge routers linked to the Internet to maintain connectivity in the event one of the upstream providers fails. Without some form of network redundancy, you’re at the mercy of a single network provider when it comes to fixing the problem.

For sites with truly massive traffic and to guarantee best performance, many organizations (such as eBay) opt for geographic redundancy. This involves creating clusters of duplicate systems that service users within designated regions, to guarantee availability as well as the fastest possible network performance. These configurations are complex and expensive to set up and run, but companies such as Cisco are now making products that midsized businesses can afford for establishing geographically distributed systems. When you need the best performance and availability, you may want to consider geographic redundancy and load balancing which is sometimes also called global load balancing.

**NOTE**

If you are hosting your Web application in-house, make sure you have a backup circuit to a network provider, in case the primary circuit becomes over-utilized or unavailable. Also, make sure you’ve got a tested action plan in place to reroute traffic if necessary.

**Install a Firewall**

Every day, Internet hackers attack both popular and unpopular Web sites. In fact, most hackers don’t target a particular site intentionally, but rather look for any vulnerable site they can use as a launching point for malicious activity. Web servers deliver information on specific ports (for example, HTTP traffic is delivered on port 80 and SSL on 443), and generally listen for connections on those ports (although you can run Web traffic on a different port if you wish). Hackers examine sites on the Internet using any number of freely available port-scanning utilities. These utilities do exactly what their name suggests: They scan points on the Internet for open ports that hackers can exploit. The best practice is to implement a front-end firewall solution, and then, if possible, place another firewall between the front-end Web servers and the database servers.

Firewalls accomplish two tasks:

- Mitigate downtime risk by examining all incoming packets, allowing only necessary traffic to reach front-end Web servers.
- Protect database and integration servers against unauthorized Internet access by allowing only communication directly from front-end Web servers.
You can build an efficient and inexpensive firewall solution using Linux's ipchains package. Red Hat Linux, for example, uses GNOME Lokkit for constructing basic ipchains networking rules. To configure specific firewall rules, however, use iptables in Red Hat (see http://www.redhat.com). For better security, the most commonly implemented front-end firewall solutions include Cisco's PIX Firewall (http://www.cisco.com), Netscreen's Firewall (http://www.netscreen.com), and Checkpoint's Firewall-1 (http://www.checkpoint.com). You must ensure that your firewall is secure as well. This means you should not run any other services on the firewall except those that are absolutely necessary.

Most vendors, including Cisco, sell load-balancing switches with built-in firewalls. The best thing to do is create a list of desired capabilities and establish a budget; then contact several vendors for quotes on affordable solutions that will meet your needs and restrictions. Be aware, too, that many modern firewall tools offer features other than port blocking. Many provide intrusion detection, intrusion alerts, blocking denial-of-service attacks, and much more.

Having a central firewall through a front-end system like the Cisco’s is a good thing, but it can lead to a false sense of security as your network is “crunchy on the outside, and soft in the middle,” meaning that if someone can get past the firewall and compromise a machine, then that person can attack with impunity because behind the firewall, all the servers can talk to each other.

To prevent this scenario, most modern OSs come with software firewalls. Local software firewalls ensure maximum security.

Use RAID Level 5 on Database Servers

Although you can build a database cluster in addition to your Web server cluster, database clusters are more complex to manage and might be impractical, depending on the size of your Web application, for your specific organization. If you have the resources for only a single database server, ensure that it is in a RAID Level 5 configuration. RAID (Redundant Array of Inexpensive Disks) stripes data across a number of disks rather than one, while reserving a separate disk to maintain CRC error checking.

Always give your transaction logs the best-performing volumes in the disk array. In any busy online transaction processing (OLTP) system, the transaction logs endure the most input/output (IO).
Disks in a RAID array are SCSI hot-swappable. If one disk in an array fails, you can substitute another in its place without affecting the server’s availability. Additionally, it is a good idea to replicate your database at regular intervals to another database server.

Another option is to use a storage area network (SAN), which is essentially a series of hard drives, allowing massive amounts of storage. SANs are highly fault tolerant and robust and allow you to not only boot multiple systems from them, but when so configured, they allow you to restore and recover a database from them in the event of a disaster scenario. One effective and relatively cheap way of adding a higher level of availability to your database layer is to use two database servers, where one is a live server that replicates the database to a SAN, and the other is a hot failover server that reads from the SAN if the primary, live server fails. This configuration provides a high level of redundancy as well as simple failover without the cost of expensive hardware, software, and database cluster management. Still, your best option is to have multiple clustered databases if you can afford it.

**Calculate Acceptable Risk**

There is always a trade-off between cost and fault tolerance. Some organizations utilize two or three Web servers configured in a cluster with a single, “strong” non-clustered database server. The database server has redundant CPUs, power supplies, disk drives, disk and RAID controllers, and network connections. This offers a good degree of availability without the additional cost of a second database server and clustering technology. Implementing a network-based tape backup strategy is another effective, cost-saving alternative and should be part of any disaster recovery plan.

Although these are reasonable risk-management approaches for some, they will be insufficient for those who need 99.999 or even 100 percent uptime. For organizations needing absolute availability, the costs and complexity of creating and managing such systems rapidly increase. If you can afford to lose a few hours or days worth of data, a simple web cluster without a database cluster is more than reasonable.

Only your budget limits the amount of redundancy you can incorporate into your system architecture. In other words, analyze your needs and plan accordingly. Any hardware can fail for virtually any reason. It is always best when arranging high availability to imagine the worst disaster and then plan based on that.

**Redundant Server Components vs. Commodity Computing**

It is recommended that you implement a fault-tolerant configuration with redundancy at every level, in order to achieve better than 99.9 percent uptime for a Web application. Most server manufacturers offer dual or triple power supplies, cooling fans, and so on in their server configurations. Choose redundant power supplies to keep servers operating in case of power supply failures. In addition, ensure that you have an uninterruptible power supply (UPS) that will power the server for a limited time in case of total power failure. Most major co-location facilities will also have their own backup generators in case of major power outages—another important consideration. In many server lines, the very low-end servers do not offer the capability to add any of these options.
Another popular approach (at Google, for instance) is to have lots of very cheap redundant servers instead of lots of redundant components. Often this arrangement is far less expensive and easier to manage—especially with recent super-low-cost blade computers—than maintaining high-end, massively redundant servers. This approach is gaining in popularity and is a major part of the emerging “grid” computing paradigm being pushed by IBM, Oracle, HP, Dell, Microsoft, and other major vendors.

Figure 46.2 shows a standard, highly available application design, including clustered Web servers, clustered database servers, Network Array Storage (NAS), redundant switches and routers, and redundant firewalls.

**Figure 46.2**
Basic high-availability site design.

**Disaster Planning**
Disaster planning and recovery processes are critical when designing and developing a high-availability system, but for some reason these needs are rarely adequately addressed. Unless your data, code, application, and hardware are not important to you, the first thing and last thing to consider is what to do when everything goes wrong. Making your system redundant and having offsite backup to prevent loss of data is not enough. Recovering from a disaster may involve rebuilding servers, applying specific patches, making tuning and configuration changes, preventing sensitive data from being exposed, as well as validating and “scrubbing” data.
Recovering from a disaster, especially one of a large magnitude, can be a daunting affair if you have not clearly and systematically addressed the recovery process. Here are some excellent resources for coming to grips with disaster recovery and planning:

- Disaster Recovery Journal (http://www.drj.com/)
- Disaster Resources (http://www.disaster-resource.com/)
- A simple Google search will reveal a wealth of tutorials, papers, and actual plans from various organizations that you can reuse to suit your specific needs.

**NOTE**

Recent laws such as Sarbanes-Oxley require organizations of a certain size to have disaster recovery plans.

When you have a plan, test it. Restore backups to make sure they work. If you aren’t in production yet, yank out power cords and network cables. An untested environment or plan has weaknesses that you haven’t discovered yet.

**Some Truths About Web Hosting**

Web site performance and availability depend as much on who hosts the site and where it’s hosted as on brilliant coding. In the last few years, hundreds of businesses have sprouted up that offer inexpensive Web hosting, but many of them do not guarantee uptime or specific service levels. When you’re designing a new Web application, you should consider the hosting question in the early design stages.

For a highly available Web site, the choice of host is important. The host can provide many features, including Internet connectivity, redundant power, backup generators, disaster recovery, on-demand bandwidth, and managed services that guarantee a 99.99 percent or greater uptime. An uptime percentage of 99.99 translates to roughly an hour of downtime per year. Choose a hosting provider that will not only guarantee this uptime but also provide some sort of reparation to you in the event that the provider fails to meet this agreement.

**NOTE**

Always choose a hosting provider that can implement an explicit service-level agreement (SLA) indicating how responsive they will be in the event of every type of site outage. Without an SLA, it’s not clear whether you or the hosting provider is responsible for recovering your application during a site outage.

**Active Site Monitoring**

ColdFusion 9 provides greatly enhanced information for monitoring site availability. But to get a true idea of how your site looks to the outside world, you should set up an active monitoring tool using another software product to collect information from outside your network. Most good ISPs and hosting providers offer some type of monitoring service, such as SolarWinds’ ipMonitor (http://www.solarwinds.com).
If you are working on your own, however, I recommend using Hewlett-Packard SiteScope, which provides a graphical dashboard of information enabling you to track and report server and site availability over days, weeks, and months. An evaluation copy of SiteScope is available from http://www.hp.com.

These types of reporting features are essential when you’re analyzing trends to create a high-availability plan for your Web application.

**TIP**

Just seeing if you can open port 80 isn’t enough—you need to implement more sophisticated server monitoring. Test for Web server health by checking specific URLs and looking for validation strings in returned Web pages.

Several other packages operate similarly to SiteScope and run on Windows, Solaris, and Linux platforms. If you are not keen on setting up and managing your own monitoring station, a few services, such as Keynote’s Performance Management Solution (http://www.keynote.com), will monitor your site from locations around the globe. Information received from your monitoring tool and these services is essential in determining and assessing availability. If your site is down due to network latency or other Internet-related issues, comparing the data produced by multiple monitoring tools or outside sources located in different locations will let you know which users couldn’t get to your site. If you notice that one network provider is consistently slow or is not meeting its uptime agreement, you should reevaluate your use of that provider.

**The Quality-of-Service Guarantee**

For high-bandwidth network transmissions, quality of service (QoS) is the idea that a network provider can predetermine and guarantee transmission rates and network quality for a client. Clients can choose a certain QoS bandwidth guarantee from a network provider, and the network will prioritize packet transmissions for that client based on a predetermined service level through the use of the Resource Reservation Protocol. This type of guarantee has become essential with the growing popularity of streaming-video multicasts. A client who plans to broadcast a high-bandwidth event at a specific date and time can contact the service provider and order the appropriate bandwidth reservation to get prioritized delivery of packets during that reservation period.

Another possible QoS guarantee may ensure 99.999 percent availability of the internal local network, individual server uptime of 99.9 percent, and clustered server uptime of 99.99 percent. The QoS guarantee ensures that your site won’t be inaccessible at a critical time.

**What Next?**

So where do we go from here? You now have a good background in understanding high availability and its benefits for your Web site. How do you implement it using ColdFusion, you ask? Chapters 47 through 51 of this book discuss various aspects of monitoring system performance, scaling with ColdFusion, and managing session state in a cluster. Understanding all these topics will aid you in building a highly available Web site running ColdFusion.
Symbols
* (asterisk)
  algorithms with, 137
  as wildcard, CFLDAP tag and,
  373–375, 378
*.docx, *.xlsx, and *.pptx extensions
  (OOXML), 469
/ (forward slash), for directory
  gateway configuration files, 561
A
absolute path for links, 55–56
acceptable risk, 13
access control, 130
accessing underlying servlet context
  (Java), 524–526
ACTION attribute
  <cfifile> tag, 317, 321
  <cfldap> tag, 370–372
ActionScript proxy classes for calling
  ESL, 550–554
Active Directory, 373–374, 386–387
active monitoring, 23–39
  basics of, 23
  Server Manager, 36–39
  Server Monitor. See Server
    Monitor
server probes, 23–26
  system probes, 26–27
ADD action (CFLDAP tag), 371
add modify type, 384
addresses
  address fields (Microsoft
    Exchange Server contacts),
    402
  Allowed IP Addresses page,
    159–160
adminapi directory (CFIDE), 225
Administrator
  accessing from Web servers,
    213–214
  Administrator page, 154–155
  Administrator Security page, 162
  configuring Web services in, 303
  enabling sandbox security via,
    189–190
  Settings Summary page, 356–357
  Administrator, securing, 153–160
  Administrator security options
    page, 153–155
  Allowed IP Addresses list,
    159–160
devolution security and, 162
  RDS security options page,
    155–156
User Manager page, 156–159
Administrator API, 233–248
  best practices, 247–248
  CFCs, 233–234
  configuring, 248
custom console coding, 248
  customized CF administration,
    235
directory, securing, 244
  façade component, 244–247
  ServiceFactory object, 242
Ajax Debug Log window, 229
alerts (Server Monitor), 34–36
algorithm parameter (encryption),
  136–137
algorithms
  Block Encryption, 137
  encoding the hash functions,
    140–142
  Hash() function, 139–140
  load-balancing, 69–70
  PBE, 137
  user-request distribution, 51
  Allowed IP Addresses page, 159–160
APIs (Server Monitor), 36
Application.cfc
  browser validation and, 145–146
  ColdFusion Login Wizard's
    Application.cfc File (listing),
    180–181
  façade component and (listing),
    245–247
  Login Wizard and, 177
Application.log, 18, 19
applications
  authentication, 164
  CFML applications for gateways,
    debugging, 567–568
deploying, 58–59
  initiator application, creating,
    565–567
  mixed Web application
    environments, scaling and,
    52
  partitioning, 86–87
  security of, 161
  simple gateway application,
    creating, 560–563
tree-tiered, 45
applications (IM), creating, 586–592
buddy lists, 587–589
gateway configuration, 591–592
gateway status management,
  590–591
  helper class, 588–592
  interactive, 592–595
messages, generating, 586–587
permissions, 591
user status management, 588–589
virtual helpdesk, users and, 587
applications, deploying
  ColdFusion Archives and. See
    ColdFusion Archives (CAR)
  ColdFusion Compiler and,
    121–125
  options for, 58–59, 109, 125
ApplicationToken attribute, 167
appointments, creating, 396–397
architecture
  event gateways, 568–569
  multi-tiered architecture with
    J2EE, 78
  physical architecture, defined, 45
tiered application architecture,
  45–46
  traditional 2-tiered Web
    architecture, 77
archives. See ColdFusion Archives
  (CAR)
ASCII files, reading and writing,
  319–320
ASP.NET Web Service
returning complex types, 458–460
returning simple types, 454–458
asynchronous processing, scaling
and, 53
Asynchronous Logging Requests,
Addition of (listing), 566–567
asynchronous printing, 363–364
asynchronous Web services, 302
Atom, 306, 310. See also feeds
attachments (Microsoft Exchange
Server), 402–403, 414–416
attributeCollection attribute, 370
attributes
<cfldap> tag, 368–369
ACTION attribute (<cfldap> tag), 370–372
<cfdirectory> tag, 336
<cfexchangecalendar> tag, 395–396
<cfexchangeconnection> tag, 392–394
<cfexchangecontact> tag, 392–394
<cfexchangefilter> tag, 395–396
<cfexchangemail> tag, 395–396
<cfexchangetask> tag, 392–394
<cfexecute> tag, 345–346
<cffeed> tag attributes, 306–307, 310–311
<cffile> tag, 317–319. See also file
systems, interacting with
<cfftp> tag, 332
<cfhttp> tag, 253–255
<cfhttpparam> tag, 256–257
<cflogin> tag, 166–167
<cfntauthenticate> tag, 179
<cfobject> tag, 449–450
<cfprint> tag, 358–360
<cfzip> tag, 348, 349
<cfzipparam> tag, 352
DN attributes, 367
gateway configuration file
attributes, 561
Java print services and, 360
MODIFYTYPE attribute, 372–373
printer-dependent, 358–360
SCOPE attribute, 372
SECURE attribute, 373
Updating Multiple (<cfldap> tag), 381
values, modifying (directories), 379–381
AttributeStruct, printer-dependent
attributes and, 359
authentication
Administrator Security page and, 162
Authenticating Users with
<cfldap> (listing), 374
basics of, 150–151, 164
with <cflogin> tag, 166–167
ColdFusion Wizard’s
Authentication File (listing), 181–183
Login Wizard and, 174–175
types of for Administrator, 154
types of for RDS, 155
authorization
Authorization Headers, 166
Authorized Users, Limiting File
Upload to (listing), 149
basics of, 150–151, 164–165
automation, Microsoft Office and, 500
B
back-end vs. front-end servers, 45–46
bandwidth, bottlenecks and, 40
bandwidth throttling, defined, 40
baselines for performance
monitoring, 17–18
Basic authentication, SharePoint Web
services and, 433
BigIP, 70
Blackbox example (sandbox security), 201–203
blogs, keeping track of. See feeds
bottlenecks
CF applications, 31, 54–56
J2EE, 80
servers, 7–8, 40–41
browser validation, 143–149
file uploads and, 148–149
form and URL hacks, 146
importance of, 143–144
Scriptprotection attribute, 145–146
techniques for, 147–148
XSS, 144–145
buddy lists, 587–589
build.xml file tool, 572
business models (Web services), 278
calendar events (Microsoft Exchange
Server), 395–400
CAR (ColdFusion Archives). See
ColdFusion Archives (CAR)
case sensitivity in Java, 504, 528
cell phones, SMS and, 595, 596–597,
602
cells (spreadsheets)
formatting, adding to, 495–500
formulas, adding to, 489–494
certificates (SSL), 411–412
<CF_UseEJB> custom tag syntax, 535
cfapplication tag, 145–146
cfcompile command-line utility, 229
cfcompile script, 121–122, 124–125
CFCs. See ColdFusion Components
(CFCs)
<cfdirectory> folder, manipulating
with. See folders, manipulating with
<cfdirectory>
CFENCODE utility, 229
CFEvent class (event gateways), 571
CFEvent object, 562–563
CFEvent objects, 558
<cfexchangecalendar> tag, 395–400
<cfexchangeconnection> tag, 392–394
<cfexchangecontact> tag, 400–403
<cfexchangefilter> tag, 403–404
<cfexchangemail> tag, 404–408
<cfexecute> tag, 345–346
<cffeed> tag attributes, 306–307, 310–311
<cffile> tag, 317–319, 324–325
<cfhttp> tag, 265–266, 267–269
<cfhttpparam> tag, 253–255
<cflogin> tag, 166–167
<cfntauthenticate> tag, 179
<cfobject> tag, 449–450
<cfprint> tag, 358–360
<cfzip> tag, 348, 349
<cfzipparam> tag, 352
dN attributes, 367
gateway configuration file
attributes, 561
Java print services and, 360
MODIFYTYPE attribute, 372–373
printer-dependent, 358–360
SCOPE attribute, 372
SECURE attribute, 373
Updating Multiple (<cfldap> tag), 381
values, modifying (directories), 379–381
CFFTP.ReturnValue variable, 270
<cfhttp> tag
<cfhttpparam> tag, 256–257
attributes, 253–255
basics of, 252
errors and results of calls, 255–256
GET method, 257–259
HTTP, using with scripting, 261–262
POST method, 262–264
queries, building from text files, 259–261
structure, 255–256
<cfhttpparam> tag, 256–257
CFID variable, 96–97, 101
CFIDE directory, securing, 225–226
CFIDE/scripts folder, 214
<cfimport> tag syntax, 523
<cfinput> tag, 147
<cfinvoke> tag, 296–297
<cfldap> tag.
See also LDAP
ACTION attribute, 370–372
basics of, 367–370
MODIFYTYPE attribute, 372–373
SCOPE attribute, 372
SECURE attribute, 373
<cflocation> tags, 97
CFLOGIN authorization variable, 168–169
<cflogin> tag, 165, 166–167
<cfloginuser> tag, 165
<cflogout> tag, 169
<cfloop> tag, reading files with, 334–335
CFML (ColdFusion Markup Language), 205–208, 529
<cfntauthenticate> tag
attributes, 179
basics of, 165, 166
fields in, 179
<cfobject> tag, 447, 502, 504
<cfprint> tag
attributes, 358, 360
basics of, 356–357
printing PDFs and, 356, 357
<cfproperty> tag, 292
CFQUERY Output from ColdFusion Web Service (listing), 464–466
<cfregistry> tag, 347
CFScript, 261–262, 275–276
cfservices.swc file, 550–551
<cfsharepoint> tag, 433–434
<cfspreadsheet> tag, 480, 481–483
CFSSOWebPart Web Part (SharePoint), 426–432
<cfthread> tag, 362–363
CFTOKEN variable, 96–97, 101
<cftry>/<cfcatch> tags, 303
CFX tags, 120
<cfzipparam> tag, 352–353
Chrome, feed rendered in, 308
ciphertext, defined, 132
circuit redundancy, 10–11
Cisco LocalDirector, 106
classes
ActionScript proxy classes for calling ESL, 550–554
classes directory (CFIDE), 225
external Java classes, using in applications, 516–520
Instantiating .NET Class (listing), 447
Java proxy class, creating, 450–452
client variables
defined, 96
vs. session variables, 101–102
storing, 103–104
clients
IM, configuration file for, 585
XMPP, 583
Cluster Manager, 66–67
clusters and clustering. See also session state in clustered environments
clustered environments, 44–45
clustered Web/database servers, 14
clustering basics, 44
ClusterManagers (JRun), 82–83
JRun, 83–85
levels of clustering for CF on JRun, 82
load-balanced clusters, 8–10, 65–68
code
coding implications of CF on J2EE, 80–81
custom console coding best practices, 248
efficient, 53–54
modular, scaling and, 53
organization of, scaling and, 52–53
ColdFusion
ColdFusion 9 portlets, 439–443
configuring for multiple hosts, 214–221
ESL, calling from, 545
extensions, 216–218
integrating Microsoft IIS servers with, 215
invoking (Web services), 296–297
running as user, 221–223
sandboxing changes in versions of, 188–189
ServiceFactory, 208
tags, risks associated with, 205–208
ColdFusion 9
accessing from portal servers, 443
accessing from SharePoint (Microsoft), 423
accessing portal servers from, 443–444
accessing Sharepoint from, 433–438
stand-alone installation, 72
support for portal servers, 438
support for SharePoint, 422
ColdFusion 9 on J2EE, development values, 74–79
built-in security features, 76–77
development ease, 76
diverse developer solutions, 76
feature-rich interfaces, 74
platform, extending, 75–76
Web site scalability, improving, 77–79
ColdFusion Administrator. See Administrator
ColdFusion Archives (CAR)
building, 113–114
CAR files, 109–110
defining, 110–113
deploying, 114–116
J2EE Archives and. See J2EE Archives
ColdFusion Bloggers, 305
ColdFusion Builder, invocation with (Web services), 298–299
ColdFusion Compiler, 121–125
ColdFusion Component Explorer, view of façade component, 239
ColdFusion Components (CFCs)
for adding new services, 75
Admin API and, 233–234
CFC That Logs Incoming CF Events (listing), 565
ColdFusion Components (CFCs)
ColdFusion Components (CFCs)  
(continued)  
CFC to Log Messages from the POP3 Gateway (listing), 578  
converting to Web services, 287–291  
LDAP CFCs, building, 388–390  
portlet CFCs, creating, 439–441  
reasons for using, 95  
ColdFusion Markup Language (CFML), 205–208, 529  
ColdFusion MX, sandbox security and, 187–188  
columns. See also query columns  
spreadsheet columns (Excel), 486  
commodity computing vs. redundant server components, 13–14  
Common Language Runtime (CLR), 445–446  
complex data types (Web services), 291–295, 297  
complex types, returning, 454, 463–467  
components  
J2EE, for extending the platform, 75  
redundant components vs. redundant servers, 14  
redundant server components vs. commodity computing, 13–14  
server and hardware components, scaling and, 46–49  
componentutils directory (CFIDE), 226  
concurrent collectors (JVM), 90–91  
Concurrent Version Control (CVS), 58  
configuring  
Admin API, 248  
alerts, 34–35  
authentication type and passwords, 154  
CF for multiple hosts, 214–221  
JRun clusters, 83–84  
load balanced clusters, 65–68  
user-based Administrator and RDS access, 156–157  
connections, 264–267, 411–414  
connectivity between CF and .NET. See .NET services, accessing probe for verifying (external), 26–27  
connector clustering (CF on JRun), 82  
constructors, Java class objects, 505–507  
consumer model (Web services), 278  
contacts (Microsoft Exchange Server), 400–403  
containers (OOXML) accessing, 470–471  
creating, 470  
Excel OOXML containers, 479–483  
Word OOXML containers. See Word OOXML containers content, probe for verifying, 24–26  
CookieDomain attribute, 167  
cookies hybrid session management and, 107  
session state management and, 94, 97–101  
Settings for Client Management with Session-Based Cookies (listing), 102  
storing login information and, 167–168  
copying files, 317, 322  
core technologies (Web services), 278–279  
cost vs. fault tolerance, 13  
CreateObject() function vs. <cfobject> tag, 504  
ServiceFactory object and, 242  
syntax for Java class objects, 502  
createObject()/<cfobject>, 296–297  
cross-site scripting (XSS), 144–145  
cryptography, 132  
Crypto-J, 137, 139–140  
CVS (Concurrent Version Control), 58  
debugging  
CFML applications for gateways, 567–568  
debugging restrictions, setting, 227–229  
print log and, 362  
Decision File, Login Wizard (listing), 181  
Decrypt() function basics of, 132–136  
parameters, 136–138  
default documents (Web servers), 232  
Delegate Access (Microsoft Exchange Server), 414  
deleting  
directories, 335  
directory entries, 384, 385–386  
Exchange items, 416  
files, 318, 322–323  
server entries, 371–372  
from zip files, 350–352  
delimited text files, building queries from, 259–261  
delivery of messages, 600–601  
deploying  
applications, 58–59  
ColdFusion Archives, 114–116  
ColdFusion templates, 123–124  
custom gateways, 578–579  
gateway interface, 570  
GatewayServices class, 570  
.NET, Java, CF, 448–449  
shared variables and, 528–529  
WSDL to CF data type conversion, 298  
database servers  
bottlenecks and, 41  
clustered, 14  
monitoring, 31–32  
Database-Driven DOC Files (listing), 478  
databases  
database connectivity, probe for verifying, 26–27  
monitoring, 31–32  
querying, 55  
storing client variables in, 102, 103–104  
tiered application architecture and, 45  
Dataset Output from .NET Web Service (listing), 461–462  
DataTable to a CF query data type example (.NET), 452–453  
Data Source, Adding (Admin API code listing), 240–241  
data tables. See tables  
data types  
CF, WSDL, and .NET, 455  
CFEvent class, 571  
complex (Web services), 291–295, 297  
DataTable to a CF query data type example, 452–453
event gateways (continued)
gateway applications, creating, 560–563
gateway instances, creating, 563–565
helper class, 572
initiator application, creating, 565–567
initiators, 559
interface, 569–570
POP3 custom gateway, 573–577
responders, 560
services class, 570
examples
Blackbox (sandbox security), 201–203
CF and .NET Web services, integration, 458
DataTable to a CF query data type (.NET), 452–453
histograms, creating, 507–516
local integration (.NET), 447–449
login, 170–173
Login Wizard, 179–183
PING, <CFEXECUTE>, Example Showing Output from (listing), 346–347
remote integration (.NET), 449–452
Unzip Example (listing), 349–350
Excel. See also spreadsheets (Excel)
    OOOXML containers, 479–483
    Exception.log, 19
    expandPath() function, 340
    exposed services, 157, 159–160

F
F5’s BigIP series, 70
façade component (Admin API), 235–241, 244–247
failover
    CF on J2EE and, 77–78
    high availability and, 8–9
    J2EE application servers and, 86 scaling and, 51–52
feeds, 305–316
    attributes, 306–307, 310–311
    basics of, 305–306
    creating, 306–310
    query columns used by, 313–315
reading, 310–316, 443
RSS and Atom basics, 306
SharePoint lists for publishing, 438
fidelity attribute, 360
fields
    in <cfntauthenticate> tag, 179
    address fields (Microsoft Exchange Server contacts), 402
CEvent object, 562–563
file systems, interacting with <cffile> tag basics, 317–319
copying/moving/re naming/deleting, 319–321
file functions, 332–335
file information, 332–333
reading and writing, 319–321
server file systems, accessing, 319–323
uploading. See uploading files
File Transfer Protocol (FTP). See FTP (File Transfer Protocol) files
    building queries from (<cfhttp>), 259–261
    configuration files (event gateways), 572
displaying with <cfftp> tag, 271–273
    Downloading Files Using <cfftp> tag with GET Method (code listings), 258, 259
downloading with <cfftp> tag, 271–273, 274
    exploring, 337–341
    file name extensions, setting, 231–232
    file operations with <cfftp> tag, 267–269
    file permissions, 186–187
    file uploads, security and, 148–149
    moving/copying/re naming, 317
    nesting and bottlenecks, 56
    reading with <cfl oop> tag, 334–335
    uploading with <cfftp> tag, 274–275
    used by Word OOOXML containers, 472
    virtual files, 341–344
    filters, 376, 403–404
    firewalls, 11–12, 14
Flash gateway adapters/security, 223–224
Flex (Adobe), 221, 549–550, 551–554
folders
    mailbox folders, querying, 394
    subfolders, exploring, 337–341
    used by Word OOOXML containers, 472
folders, manipulating with <cfdirectory>, 335–341
directory contents, obtaining, 337
directory explorer, building, 337–341
Listing Files and Folders Within Directories (listing), 338–340
    using <cfdirectory>, 335–337
form variables, 96–97
formatting cells, 495–500
forms. See also HTML forms
    Cold Fusion Login Wizard’s Login Form (listing), 180
    control over file uploads with form fields, 330
    form hacks, 146
    Sample Login Form (listing), 171–172
    formulas, adding to cells, 489–494
    Forta, Ben, 361
    forward slash (/), for directory
    gateway configuration files, 561
    Framework Class Library (FCL), 446–447
    Fresh feed reader, 305
    front-end vs. back-end servers, 45–46
    FTP (File Transfer Protocol). See <cfftp> tag functions
    CF encryption functions, 131–132
    file functions, 332–335
    security functions, 165–166
    spreadsheets (Excel), 481
Fusebox, 53

G
Gamma, Erich, 235
“Gang of Four”, 235
garbage collection (JVM), 87, 88–91
Gates, Bill, 445
gateway adapters/security (Flash), 223–224
gateway-config.xml, 223
gateways. See event gateways
GenerateSecretKey function, 132, 133, 136

generations (JVM), defined, 89
geographic redundancy, 11
GET method, 257–259, 262
GetAuthUser() function, 165, 172–173
getFileInfo() function, 333–334
GetHostName() method, 447
getHTTPRequestData() function, 263
GetNames method, 459, 460, 464
GetPageContext() function, 524–525
global load balancing, 11
GUI tool, creating Java proxy class with, 450–452
hardware defined, 43
hardware vs. software load-balancing options, 62–64
hardware-based load balancing, 9–10, 68–70
hardware-based session management, 106
hardware components, scaling and, 46–49
Hash() function, 138–143
basic algorithms, 139–140
Distributed mode, 143
encoding the hash functions algorithm, 140–142
page requests and SSL, 142–143
SSL liabilities, 143
Helm, Richard, 235
helpdesk application. See IM (instant messaging) gateways; SMS gateways
helper class (event gateways), 572
Hewlett-Packard SiteScope, 16
high availability, 3–16
acceptable risk, 13
active site monitoring, 15–16
basics of, 3–4
circuit redundancy, 10–11
disaster planning, 14–15
firewalls, 11–12
hosting, 15
load-balanced clusters, 8–10
planning for, 8
QoS guarantees, 16
RAID Level 5, 12–13
redundant components, 13–14
redundant servers, 14
server load and, 5–8
histograms, creating (example), 507–516
hops, defined, 10
hosting, performance and availability and, 15
hot deploy techniques, 79
hot spare, defined, 64
HPtune, 92
HTML forms
hacking and, 146
HTML Form for File Upload Using <cfimage> (listing), 326–327
Modification of HTML to Show Data-Driven Attribute Setting (listing), 329–330
HTTP (Hypertext Transfer Protocol), 252, 261–262, 279.
See also <cfhttp> tag
IdleTimeout attribute, 166
IIS Lockdown (Microsoft), 144
IIS servers (Microsoft), 215
IM (instant messaging) gateways, 581–595
applications, creating. See applications (IM), creating basics of, 581–583
configuration file, 585
defining, 583–585
development environment and, 583
interactive applications, creating, 592–595
presence, 582–583
XMPP clients, 583
images (inline), retrieving, 416
importing JSP custom tag libraries, 523
incremental collectors (JVM), 90, 91
infrastructure defined, 43
legacy infrastructure, support for, 73–74
J2EE (Java 2 Enterprise Edition). See also scaling with J2EE
modularity of component design, 80
session management, 80–81, 104–105, 107
J2EE Archives
defining, 116–120
deploying, 120–121
overview, 116
memory use, 19, 30–31
Merchandise Order spreadsheet, 496
messages, 586–587, 595, 600–601
metadata properties, iTunes extensions and, 316
methods
   Call a Remote Method (listing), 452
   CFEvent class, 571
   Exercise a .NET Method (listing), 447
   gateway interface, 570
   GatewayServices class, 570
   get methods, exposing, 466
   IM gateway, 584–585
   List of Methods (listing), 460–461
   PageContext, 525
metrics (JRun), 21–22
Microsoft
   IIS Lockdown, 144
   Log Parser 2.2, 20
   Visual SourceSafe and Robocopy, 58
   Windows, processor performance in, 47
Microsoft Exchange Server, 391–419
   <cfexchangecalendar> tag, 395–400
   <cfexchangeconnection> tag, 392–394
   <cfexchangemessage> tag, 400–403
   <cfexchangefilter> tag, 403–404
   <cfexchangelist> tag, 404–408
   <cfexchangetask> tag, 408–411
   connections, managing, 411–414
   exchange items, managing, 414–419
   Exchange Server tags basics, 391
Microsoft IIS servers, CF and, 215
Microsoft Office, integrating with, 469–500
   automation, 500
dynamic Excel OOXML spreadsheets, creating, 480–483
Excel OOXML containers, 479–483
OOXML basics, 469–471
OOXML Word document files (containers), 470–471
spreadsheets, adding formatting to cells, 495–500
   spreadsheets, adding formulas to cells, 489–494
   spreadsheets, creating, 483–484, 487–489
   spreadsheets, populating, 485–488
   spreadsheets, reading, 484–485
Word OOXML containers. See
   Word OOXML containers
mm_wizard_application_include.cfm file, 177
mm_wizard_authenticate.cfc file, 177, 181–183
mm_wizard_login.cfm file, 178, 180
MODIFY action (<cfldap> tag), 371
MODIFYTYPE, 372–373, 383–384
modular code, 53
modularity, scaling and, 80
monitoring. See also active monitoring; system performance, monitoring
   site availability, 15–16
moving files, 317, 322
multi-homed Web servers, 212–214
multiple hosts, configuring ColdFusion for, 214–221
multiple server configurations, securing, 211–214
multithreading, scaling and, 53
name conventions (LDAP), 367
   nameConflict Values for <cffile> action="Upload", 324–325
   native load balancing, 78–79
nesting files and bottlenecks, 56
.NET, 445–467
calling ESL from, 546–548
   CF Web service, returning a complex type, 463–467
   fundamentals, 445–446
   .Net Web service, returning a complex type, 458–463
   .Net Web service, returning a simple type, 454
   .NET Web services, invoking, 299
.NET services, accessing basics of, 446
   DataTable to a CF query data type example, 452–453
local integration example, 447–449
   remote integration example, 449–452
NewSize attribute (JVM), 90
   nillable arguments, calling Web services with, 301–302
object clustering (CF on JRun), 82
   objectclass property (Active Directory), 386–387
   objects, spreadsheets and, 481–483, 488–489
Office Open XML (OOXML), 469–471
   Offline AIR file path permissions, 201
   onIncomingMessage() function, 600, 601
   OOXML (Office Open XML), 469–471
   OpenOffice file path permissions, 201
   operating systems, interacting with, 317
   OptimizeIT, 87
packet sniffers, 131
Page Viewer Web Part (SharePoint), 423–425
PageContext methods, 525
Pages, Retrieving via <cfhttp> (listing), 257
Paragraphs, Adding to Blank Word 2007 File (listing), 474
   parsing feeds, 312
   Password-Based Encryption (PBE).
      See PBE (Password-Based Encryption)
   passwords
      Administrator page for, 154–155
      Administrator Security page and, 162
      CF Admin security and, 243–244
      facade component and, 244–247
      Login Wizard and, 175
      RDS page and, 155–156, 162–163
      RDS passwords (Admin API), 248
      User Manager page and, 157–159
Index

path caching, Web server, 212–213
PBE (Password-Based Encryption), 133, 137, 138
PDFs, 355–356, 363
percentage load, defined, 5
performance. See also system performance, monitoring performance testing, load and, 5–8
perm generations (JVM), 89
permissions
file and directory, 186–187
Offline AIR file path, 188–189
OpenOffice file path, 201
printer, 362
virtual helpdesk, 591
persistent connections (Microsoft Exchange Server), 412–413
persistent option for managing session variables (J2EE), 81
physical architecture, defined, 45
PING, <CFEXECUTE> Example Showing Output from (listing), 346–347
placeholders(#), Replacing with Dynamic Content (listing), 476–477
platforms, extending, 75–76
podcasts, support with feeds, 314–316
POP3 custom gateway, 573–577
populating spreadsheets, 485–488
portal servers
accessing ColdFusion 9 from, 443
accessing from ColdFusion 9, 443–444
ColdFusion 9 support for, 438
Java, 422, 443
using remote ColdFusion 9 portlet with, 442
portlets, 439–443
port-scanning utilities, 11–12
portType and operation from WSDL (listing), 283
POST method
</cfhtto> tag, 253
<cfhttp> tag, 253, 262–264
vs. GET method, 262
precompiling ColdFusion templates, 122–123
presence (IM gateways), 582–583
principals (Microsoft Exchange Server), 414
print log, 362
printers
capabilities, controlling, 358–360
determining available, 361–362
permissions, 362
printing. See also server-side printing to default/non-default printers, 357
probes
server probes, 23–26
system probes, 26–27
programs on servers, executing, 345–347
properties
calendar events (Microsoft Exchange Server), 397–398
contacts (Microsoft Exchange Server), 402
message property values (Microsoft Exchange Server), 408
provider model (Web services), 278
Q
quality of service (QoS) guarantees, 16
queries
building from text files (cfhttp), 259–261
database monitoring and, 31
DataTable to a CF query data type example (.NET), 452–453
Running Simple for DN (listing), 379
writing database queries, 55
query action (<cfldap> tag), 370–371
query columns
<cfexchangeconnection> getSubfolders, 394
<cffeed> tag, 312–314
<cfzip> tag, 350
associated with Dublin Core extensions, 315
associated with iTunes extensions, 316
populated by <cfdirectory> action="List", 337
query object definitions, <cftrp> tag, 273
querying databases, 55
querying directories, 374–377

R
RAID (Redundant Array of Inexpensive Disks)
Level 5, high availability and, 12–13
scaling considerations and, 48
RAM, scaling considerations and, 47–48
RDS page, security and, 155–156, 162–163
RDS passwords (Admin API), 248
reading
feeds, 310–316
files, 318, 319–321, 334–335
spreadsheets, 484–485
zip files, 350–352
Recurrence types (Microsoft Exchange Server calendar events), 398–399
redundant servers vs. redundant components, 14
Registry, storing client variables and, 101
relative path for links, 55–56
Remote Development Service (RDS), securing, 208–210
remote server integration example (.NET), 449–452
renaming directories and files, 317, 323, 335
replace modify type, 384
reports
database reports, 31–32
server statistics, 32–34
REQUEST variables, sharing, 527–529
Request-response operation type (WSDL), 284
resource security, 185
responders, 560–565
responding to messages, 600
retrieving Exchange items and attachments, 414–416
Robocopy, 58
round robin load balancing method, 69–70
rows (Excel), 486
RRDNS (round-robin DNS), load balancing and, 49–50
RSA BSafe Crypto-J library algorithms, 139–140
RSA JSafe Crypto libraries, 114
RSS (Really Simple Syndication), 306. See also feeds
sandbox security
applying, 210–211
basics of, 185–190
Blackbox example, 201–203
ColdFusion MX and, 187–188
creating and securing applications with, 190
enabling, 191–192
enabling via CF Admin, 189–190
sandboxes, adding, 193
sandboxes, configuring, 194–200
sandboxes, tips for creating, 200–201
sandboxes, 157, 244
scaling, 43, 44–45
scaling with CF, 43–70
applications that scale, writing, 52–56
failover, 51–52
hardware load-balancing options, 68–70
hardware vs. software load-balancing options, 62–64, 69
importance of, 43–44
load-balancing options, selecting, 49–50
mixed Web application environments, 52
multiple instances, 59–62
server and hardware components, 46–49
servers, keeping in sync, 57–59
session state management, 51
tiering, 45–46
user-request distribution algorithms, 51
using load balancing, 65–68
scaling with J2EE, 71–92
basics of, 71
CF and JRun, 82–85
CF on J2EE platforms, 72–74
CF on top of Java servers, 79–81
development values of CF on J2EE. See ColdFusion 9 on J2EE, development values
improving Web site scalability, 77–79
JVM, tuning. See Java Virtual Machine (JVM), tuning other Java application servers, 85–86
SCOPE attribute, 372
scripting using FTP with, 275–276
using HTTP with, 261–262
Scriptprotect attribute, 145–146
searching directories, 366, 372
search filters (<cfldap> tag), 376
SECURE attribute, 373
Secure Socket Layers (SSL) basics for security, 129
certificates, 411–412
liabilities, 143
page requests and, 142–143
security. See also Administrator, securing
Active Directory and, 373–375
authentication and authorization, 150–151
browser validation. See browser validation
CF concerns and, 129–131
development security, 162–163
embedding in URL strings and, 97
encryption. See encryption;
Hash() function
J2EE built-in security features, 76–77
LDAP calls, 373
Login example, 170–173
Login Wizard, 173–178
Login Wizard example, 179–183
secure FTP connections, 275
security framework basics, 161–162
security risks, 129–131
user security. See user security security in shared/hosted environments
CF for multiple hosts, configuring, 214–221
CF ServiceFactory and, 208
CFIDE directory, securing, 225–226
CFML-based risks, 205–208
default documents (Web server), 232
encrypting ColdFusion templates, 229–230
error messages, 229–230
Flash gateway adapters/security, 223–224
JSP functionality, disabling, 225
multiple server configurations, securing, 211–214
RDS, securing, 208–210
removing Help docs and example applications, 227
running ColdFusion as user, 221–223
sandbox security, applying, 210–211. See also sandbox security
session timeout values, limiting, 227
setting custom tag paths, 230–231
setting debugging restrictions, 227–229
setting file name extensions, 231–232
vendor security update sites, 232
sendGatewayMessage function, 559, 586–587
serialization of CFCs, 105–106
server components, scaling and, 46–49
server heartbeat, defined, 9
server load, 5–8, 52, 86
Server Manager, 36–39
Server Monitor, 27–36
alerts, 34–36
API, 36
basics of, 27–29
database monitoring, 31–32
error monitoring, 32
performance problems and, 40
server statistics, 32–34
snapshots, 35
starting, 29–30
tracking memory use, 30–31
server probes, 23–26
Server.log, 19–20
servers. See also database servers;
Web servers
comparisons between, 39
connector clustering, 82–83
creating new (JRun), 82
directing client to most available, 102
dual-homed servers, 46
duplicating settings, 37–39
front-end vs. back-end servers, 45–46
servers (continued)
  keeping in sync, 57–59
  local and remote connections, 446
  local server integration example, 447–449
  multiple server configurations, securing, 211–214
  programs on, executing, 345–347
  remote server integration example, 450–452
  server's file system, accessing, 319–323
  statistics, 32–34
  uploading files from, 324–325
server-side printing, 355–363
<cfprint> attributes, 358
<cfprint> basics, 357–358
<cfthread> tag, 362–363
basics of, 355–356
print log, 362
printer capabilities, controlling, 358–360
printers, determining available, 361–362
printing other than PDFs, 363
reducing waiting time, 363–364
server-side variables, 96
ServiceFactory (ColdFusion), 208
ServiceFactory object (Admin API), 242
services class (event gateways), 570
servlets, Java. See Java servlets
session state in clustered environments, 93–107
client variables, storing, 103–104
cookies, 94, 97–101
hardware-based management, 106
history of, 94
hybrid solutions, 106
Java session management, 104–105
options for, 95
serializing, 105–106
session state, defined, 93
session variables basics, 95–96
session vs. client variables, 101–102
URL parameters or form variables, 96–97
users and servers, 102
session state management, scaling and, 51, 80–81
session timeout values, limiting, 227
session variables
  basics of, 95–96
  vs. client variables, 101–102
  defined, 96
  hybrid session management and, 107
  passing from page to page, 96–97
  persistent option and (J2EE), 81
  sticky option for (J2EE), 81
  URL parameters, 96–97
SESSION.SESSIONID variable, 105
SharePoint (Microsoft)
  accessing CF 9 data/content from, 432–433
  accessing ColdFusion 9 from, 423
  accessing from ColdFusion 9, 433–438
  CFSSOWebPart Web Part to access CF 9 applications, 426–432
  ColdFusion 9 support for, 422
  Page Viewer Web Part to access CF 9 applications, 423–425
  Web services, consuming with CF 9, 433–438
<show-stacktraces> tags, 224
SMS (Short Message Service)
  gateways, 595–602
  basics of, 595–597
  defining, 597–599
  extending, 601–602
  messages, generating, 599
  messages, responding to, 600–601
  testing, 596
SMS gateways, 581
snapshots (Server Monitor), 35
SOAP (Simple Object Access Protocol), 279, 300–302
software
  defined, 43
  hardware vs. software load-balancing options, 62–64
  software-based load balancing, 9, 10
sourceless templates, compiling, 121–122
spreadsheets (Excel)
  <CFSPREADSHEET> tag, 481–483
  creating, 480–484, 488–489
  formatting, adding to cells, 495–500
  formulas, adding to cells, 489–494
functions, 481
populating, 485–488
reading, 484–485
spreadsheet objects, creating, 481
SpreadsheetAddRows function, 483
SQL injection or SQL poisoning, 146
SSL. See Secure Socket Layers (SSL)
SSO (single sign-on) feature (SharePoint), 423, 426
stand-alone Web servers, 212–213
standards, Java and CF and, 72
statistics, usage statistics for Web sites, 6
sticky option, session variables and (J2EE), 81
storage area network (SAN), defined, 13
stress testing, defined, 5
string parameter (encryption), 136, 139
strings, 282–283, 480
struct data type, 290–291
subfolders
  exploring, 337–341
  mailbox folders, querying, 394
Sun Java system Web server, 218–221
SunJCE provider, algorithms support by, 137–138, 139
survivor spaces (JVM), 89
syndication model (Web services), 278
system performance, monitoring, 17–41
active monitoring. See active monitoring
baselines, 17–18
historical analysis, 18
JRun logs, 20–22
log files, analyzing, 18–20
performance issues and options, 40
performance vs. readability/modularity/maintainability, 53
typical bottlenecks, 40–41
Web server log files, analyzing, 22–23
system probes, 26–27
system registry, interacting with, 347
tables, 452–453
tag paths, setting custom, 230–231
taglibs (JSP), 522–524
tags. See specific tags
tasks (Microsoft Exchange Server), 408–411
templates
  compiling sourceless, 121–122
  creating Word OOOXML documents from, 476, 479
deployment of, 123–124
  encrypting, 229–230
  precompiling, 122–123
  sourceless, 121–122
tenured generations (JVM), 89
testing
  JVM applications, 91–92
  load performance, 5
  SMS applications, 596, 599
  stress testing, defined, 5
text files, 259–261, 319–320
text messages, 595, 600–601
textQualifier attribute, 260
threads, multithreading, scaling
  and, 53
three-tiered applications, 45
throughput collectors (JVM), 90
Thumbnails of New Images, CFC that Creates (listing), 562
tiering
  defined, 45
  tiered application architecture, 45–46
  tiered infrastructure with CF on J2EE, 77–78
  tradition 2-tiered Web architecture, 77
timeout values, limiting (sessions), 227
tools. See also Server Monitor
development tools (event gateways), 572
  GUI tool, creating Java proxy
class with, 450–452
  JavaLoader tool, 520
  URLScan Security tools, 144
  Web Server Configuration Tool, 67–68
  wsconfig tool, 219–221
  WSDL2 Java tool, 299
transfer protocols, 251. See also
  <cfhttp> tag; <cfhttp> tag; FTP
(File Transfer Protocol); HTTP (Hypertext Transfer Protocol)
transient connections (Microsoft Exchange Server), 392, 413
types. See also complex types, returning; data types
event types (event gateways), 578 .NET, 448–449
simple types, returning (.Net Web service), 454
Universal Naming Convention (UNC), 56
Unix, processor performance in, 47
Unzip Example (listing), 349–350
uploading files
  with <cffile> tag, 318
  with <cfhttp> tag, 258, 274–275
  basics of, 323–324
  multiple uploads, 331–332
  security and, 148–149
  status of file uploads, 325–326
  that exist on server, 324–325
  upload interface, 326–331
URL hacks, 146, 147
URL parameters in session state management, 96–97
URLScan Security tools, 144
usage statistics for Web sites, 6
User Manager page, 156–159
user security, 163–169
  authenticating with <cflogin> tag, 166–167
  authentication and, 164
  authorization and, 164–165
  CFLOGIN authorization variable, 168–169
  logging out, 169–170
  login information, storing, 167–168
  tags and functions, 165–166
user sessions, defined, 93
user-request distribution algorithms, 51
users
  Authenticating Users with <cfldap> (listing), 374
  buddy lists, 587–588
  creating for ESL, 541–543
  creating new (User Manager page), 157–159
  running ColdFusion as, 221–223
  text messages to multiple, 599
validation. See browser validation
  attributes, modifying, 379–381
  of CFFTP.ReturnValue variable, 270
  format struct name and value pairs, 495–496
  message property values, 409
  nameConflict Values for <cffile> action="Upload", 324
  for scriptProtect, 146
variables. See also client variables;
session variables
  application variables, 96
  available after file uploads, 325–326
  CFFTP, 269
  form variables, 96–97
  passed through POST method, 262, 263–264
  session state management and, 96
  sharing among CF/JSP/servlets, 527–529
  values of CFFTP.ReturnValue variable, 270
Verity mappings, 120
virtual files, 341–344
virtual helpdesk. See IM (instant messaging) gateways
virtual private network (VPN), 143
Visual SourceSafe, 58
Visual Studio, building ASP.NET Web Service in, 454–458
Vlissides, John, 235
WAR files
  defined, 85
  J2EE Archives and, 116
Web hosting, hosting, performance and availability and, 15
Web portals, 422–423
Web server clustering (CF on JRun), 82
Web Server Configuration Tool, 67–68
Web servers. See also servers
accessing CF Admin from, 213–214
authentication, 164
clustering (CF on JRun), 82
clusters, adding to (JRun), 82, 83
multi-homed, 212–214
path caching, 212–213

Web services
ASP.NET Web Service, building, 454–458
business models, 278
calling with nillable arguments, 301–302
CF, invoking, 296–297
CF and .NET, integration example, 458
complex data types, 291–295, 297
configuring in CF Admin, 303
consuming, 295
converting CFCs to, 287–291
core technologies, 278–279
creating, 286
error handling and, 303
ESL and, 539–541
vs. intelligent agents, 251
invocation with ColdFusion Builder, 298–299
invoking .NET Web services, 299
overview, 277–278
programming best practices, 302–303
returning complex types (CF), 463–467
returning complex types (.NET), 458–463
returning simple types, 454
SharePoint and, 432
SOAP requests and, 300–302

Web sites
improving scalability, 77–79
monitoring availability, 15–16

Web sites for downloading
CFSSOWebPart Web Part, 429
Fresh feed reader, 305
Hewlett-Packard SiteScope, 16
JavaLoader tool, 520
JCE Unlimited Strength Jurisdiction Policy Files 6, 132
Login Wizard extension, 173
Microsoft Log Parser 2.2, 20
Server Manager installers, 37
tag for printer information, 361
Visual Studio Express editions, 445
Web sites for further information
AuditMyPC.com, 12
CF 9 security lockdown guide, 221
CFMX IM Event Gateway, 582
ColdFusion Bloggers, 305
COM callable wrapper-based integration, 447
disaster recovery, 15
firewall solutions, 12
Fusebox, 53
integrating CF and .NET, 467
Java Cryptography Architecture Sun Providers
Documentation, 133
Java keytool command, 412
Java stack traces, 19
JavaDoc for Properties class, 572
JBuddyCF, 582
JProfile, 87
LDAP, 365
load balancing solutions for Linux, 9
load-testing products, 5–6
OptimizeIT, 87
SharePoint SSO feature, 426
site monitoring services, 16–17
tag libraries, 522
URLScan Security tools, 144
vendor security update sites, 232
XMPP, 582
XXS, 144
WebLoad, 18
<whitelist> entry, 224
wizards directory (CFIDE), 226
Word OOXML containers
basics of, 471–472
creating, 473–475
dynamic Word OOXML documents, creating, 475–479
viewing, 475
worksheets
Expenses worksheet, 489
Films worksheet, 488
Merchandise worksheet, 492
writing
applications that scale, 52–56
database queries, 55
files, 319–321
wsconfig tool, 219–221, 231–232

WSDL (Web Services Description Language)
document elements, 280
generating, 299
message and part tags from (listing), 283–284
number-to-string conversion Web service (listing), 282–283
operation types in, 284
Output from NameServices.cfc (listing), 464
overview, 279–280
portType and operation from (listing), 283
sample layout, 281
SharePoint Web service and, 433–434
simple Web service (listing), 284–286
WSDL to ColdFusion data type conversion, 298
WSDL2 Java tool, 299

WSRP
Consumer Web Part, 433
Consumers (portal servers), 439
v1 specification, 438–439

X
XML (Extensible Markup Language)
declared, 279
XMLSearch function, 475
XMPP (Extensible Messaging and Presence Protocol), 582, 583, 584
XSS (cross-site scripting), 144–145

Y
young generations (JVM), 89

Z
zip files, 347–353
<cfzipparam>, 352–353
basics of, 347–348
creating, 348–349
expanding, 349–350
listing/reading/deleting from, 350–352
This page intentionally left blank
Meet Creative Edge.

A new resource of unlimited books, videos and tutorials for creatives from the world’s leading experts.

Creative Edge is your one stop for inspiration, answers to technical questions and ways to stay at the top of your game so you can focus on what you do best—being creative.

All for only $24.99 per month for access—any day any time you need it.

peachpit.com/creativeedge