WebSphere Product Family Overview and Architecture

Discover the WebSphere family

Take an in-depth look at key products

Compare capabilities

Carla Sadtler
Gennaro Cuomo
John Ganci
Marc Haberkorn
Carol Jones
Peter Kovari
Kevin Griffith
Dildar Marhas
Rob Will

ibm.com/redbooks
Note: Before using this information and the product it supports, read the information in “Notices” on page xv.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.11 Administration</td>
<td>cxvii</td>
</tr>
<tr>
<td>2.11.1 Administration tools</td>
<td>cxviii</td>
</tr>
<tr>
<td>2.11.2 Configuration repository</td>
<td>cxix</td>
</tr>
<tr>
<td>2.11.3 Centralized administration</td>
<td>cxix</td>
</tr>
<tr>
<td>2.12 The flow of an application</td>
<td>cxxi</td>
</tr>
<tr>
<td>2.13 Developing and deploying applications</td>
<td>cxxiii</td>
</tr>
<tr>
<td>Chapter 3. IBM WebSphere Extended Deployment</td>
<td>cxxvii</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>cxxviii</td>
</tr>
<tr>
<td>3.2 An adaptable IT infrastructure</td>
<td>cxxix</td>
</tr>
<tr>
<td>3.3 Dynamic operations</td>
<td>cxxx</td>
</tr>
<tr>
<td>3.3.1 Resource sharing</td>
<td>cxxxi</td>
</tr>
<tr>
<td>3.3.2 Application service levels</td>
<td>cxxxii</td>
</tr>
<tr>
<td>3.3.3 Performance requirements</td>
<td>cxxxiii</td>
</tr>
<tr>
<td>3.3.4 Focus on business strategy - not the technology</td>
<td>cxxxiv</td>
</tr>
<tr>
<td>3.3.5 Autonomic capabilities</td>
<td>cxxxvi</td>
</tr>
<tr>
<td>3.3.6 Scale beyond your defined environment</td>
<td>cxxxvii</td>
</tr>
<tr>
<td>3.4 Extended manageability</td>
<td>cxxxvii</td>
</tr>
<tr>
<td>3.4.1 View your infrastructure run-time status</td>
<td>cxxxviii</td>
</tr>
<tr>
<td>3.4.2 Know when intervention is required</td>
<td>cxlii</td>
</tr>
<tr>
<td>3.4.3 View the inner workings of your infrastructure</td>
<td>cxlii</td>
</tr>
<tr>
<td>3.5 High volume transaction requirements</td>
<td>cxliii</td>
</tr>
<tr>
<td>3.5.1 Partitioning pattern</td>
<td>cxliii</td>
</tr>
<tr>
<td>3.5.2 Partitioning</td>
<td>cxlv</td>
</tr>
<tr>
<td>3.5.3 Application partitioning using the WebSphere partition facility</td>
<td>cxlvi</td>
</tr>
<tr>
<td>3.5.4 Partition-aware workload management</td>
<td>cxlvi</td>
</tr>
<tr>
<td>3.5.5 Leveraging partitions</td>
<td>cxlvii</td>
</tr>
<tr>
<td>3.5.6 Recover from failures quickly</td>
<td>cxlviii</td>
</tr>
<tr>
<td>3.5.7 Rebalance partitions</td>
<td>cxlxi</td>
</tr>
<tr>
<td>3.6 Summary</td>
<td>cxlxi</td>
</tr>
<tr>
<td>3.7 For more information</td>
<td>cl</td>
</tr>
</tbody>
</table>

Chapter 4. WebSphere Application Server V6 technical overview  
4.1 Application server configurations          
4.1.1 Stand-alone server configuration         
4.1.2 Distributed server configuration        
4.2 Application servers, nodes, and cells      
4.2.1 Application servers                      
4.2.2 Nodes, node groups, and node agents      
4.2.3 Cells                                    
4.3 Servers                                   
4.3.1 Application servers                      
4.3.2 Clusters                                 

Contents v
4.3.3 JMS servers (V5) ................................................ clvii
4.3.4 External servers ................................................. clvii
4.4 Containers ......................................................... clviii
  4.4.1 Web container ................................................ clviii
  4.4.2 Enterprise JavaBeans container ........................ clx
  4.4.3 Application client container ............................. clx
4.5 Application server services ................................... clxi
  4.5.1 J2EE Connector Architecture services ................. clxii
  4.5.2 Transaction service ........................................ clxii
  4.5.3 Dynamic cache service .................................... clxiii
  4.5.4 Message listener service ................................ clxiv
  4.5.5 Object Request Broker service .......................... clxiv
  4.5.6 Administrative service .................................. clxv
  4.5.7 Name service .............................................. clxv
  4.5.8 Performance Monitoring Infrastructure service .... clxvii
  4.5.9 Security service .......................................... clxviii
4.6 Data Replication Service ...................................... clxviii
4.7 Virtual hosts. ................................................... clxix
4.8 Session management ........................................... clxx
  4.8.1 HTTP Session persistence ................................ clxxi
  4.8.2 Stateful session EJB persistence ....................... clxxii
4.9 Web services .................................................... clxxii
  4.9.1 Enterprise services (JCA Web services) .............. clxxiv
  4.9.2 Web service client ........................................ clxxv
  4.9.3 Web service provider .................................... clxxv
  4.9.4 Enterprise Web Services ................................ clxxv
  4.9.5 IBM WebSphere UDDI Registry ........................... clxxvi
  4.9.6 Web Services Gateway ................................... clxxvi
4.10 Service integration bus ...................................... clxxviii
  4.10.1 Application support ..................................... clxxx
  4.10.2 Service integration bus and messaging ............... clxxx
  4.10.3 Web services and the service integration bus .... clxxxii
4.11 Security ......................................................... clxxxiii
  4.11.1 User registry ............................................. clxxxv
  4.11.2 Authentication ........................................... clxxxvi
  4.11.3 Authorization ............................................. clxxxvii
  4.11.4 Security components .................................... clxxxviii
  4.11.5 Security flows ............................................ clxxxix
4.12 Resource providers ........................................... cxc
  4.12.1 JDBC resources ........................................... cxc
  4.12.2 Mail providers ............................................. cxcii
  4.12.3 JCA resource adapters ................................ cxciii
  4.12.4 URL providers ........................................... cxciv
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4.2</td>
<td>Struts Configuration File Editor</td>
<td>cccxxxvi</td>
</tr>
<tr>
<td>7.5</td>
<td>JavaServer Faces tools</td>
<td>cccxxxvi</td>
</tr>
<tr>
<td>7.6</td>
<td>XML tools</td>
<td>cccxxxvi</td>
</tr>
<tr>
<td>7.6.1</td>
<td>XML Editor</td>
<td>cccxxxvii</td>
</tr>
<tr>
<td>7.6.2</td>
<td>Document Type Definition (DTD) Editor</td>
<td>cccxxxvii</td>
</tr>
<tr>
<td>7.6.3</td>
<td>XML Schema Editor</td>
<td>cccxxxvii</td>
</tr>
<tr>
<td>7.6.4</td>
<td>Extensible Style Sheet Language (XSL) Editor</td>
<td>cccxxxvii</td>
</tr>
<tr>
<td>7.6.5</td>
<td>XPath Expression wizard</td>
<td>cccxxxvii</td>
</tr>
<tr>
<td>7.6.6</td>
<td>XSL debugging and transformation</td>
<td>cccxxxvii</td>
</tr>
<tr>
<td>7.6.7</td>
<td>XML and SQL query</td>
<td>cccxxxviii</td>
</tr>
<tr>
<td>7.6.8</td>
<td>XML to XML Mapping Editor</td>
<td>cccxxxviii</td>
</tr>
<tr>
<td>7.6.9</td>
<td>Relational database to XML mapping</td>
<td>cccxxxviii</td>
</tr>
<tr>
<td>7.7</td>
<td>Relational database tools</td>
<td>cccxxviii</td>
</tr>
<tr>
<td>7.7.1</td>
<td>Databases and schemas</td>
<td>cccxxix</td>
</tr>
<tr>
<td>7.7.2</td>
<td>Tables and views</td>
<td>cccxxix</td>
</tr>
<tr>
<td>7.7.3</td>
<td>SQL statements</td>
<td>cccxxix</td>
</tr>
<tr>
<td>7.7.4</td>
<td>DB2 routines</td>
<td>cccxxix</td>
</tr>
<tr>
<td>7.7.5</td>
<td>SQLJ files</td>
<td>cccxl</td>
</tr>
<tr>
<td>7.8</td>
<td>Java development tools (JDT)</td>
<td>cccxl</td>
</tr>
<tr>
<td>7.8.1</td>
<td>Java perspectives</td>
<td>cccxl</td>
</tr>
<tr>
<td>7.8.2</td>
<td>Java Editor</td>
<td>cccxli</td>
</tr>
<tr>
<td>7.8.3</td>
<td>Java builder</td>
<td>cccxlii</td>
</tr>
<tr>
<td>7.8.4</td>
<td>Refactoring support</td>
<td>cccxlii</td>
</tr>
<tr>
<td>7.8.5</td>
<td>Debugger</td>
<td>cccxlii</td>
</tr>
<tr>
<td>7.8.6</td>
<td>Scrapbook</td>
<td>cccxliv</td>
</tr>
<tr>
<td>7.8.7</td>
<td>Java search</td>
<td>cccxliv</td>
</tr>
<tr>
<td>7.8.8</td>
<td>Javadoc</td>
<td>cccxliv</td>
</tr>
<tr>
<td>7.8.9</td>
<td>Visual Editor for Java</td>
<td>cccxliv</td>
</tr>
<tr>
<td>7.9</td>
<td>Enterprise JavaBean (EJB) tools</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.9.1</td>
<td>J2EE perspective</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.9.2</td>
<td>Creating enterprise beans and access beans</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.9.3</td>
<td>Creating access beans</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.9.4</td>
<td>Building data persistence into enterprise beans</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.9.5</td>
<td>Generating deployment code</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.9.6</td>
<td>Validating enterprise beans and access bean code</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.9.7</td>
<td>EJB and Java UML visual editing</td>
<td>cccxlvi</td>
</tr>
<tr>
<td>7.10</td>
<td>Web services development tools</td>
<td>cccxlviii</td>
</tr>
<tr>
<td>7.10.1</td>
<td>Creating Web services</td>
<td>cccxlxi</td>
</tr>
<tr>
<td>7.10.2</td>
<td>Using Web services (developing clients)</td>
<td>ccccli</td>
</tr>
<tr>
<td>7.11</td>
<td>Portal tools</td>
<td>ccclii</td>
</tr>
<tr>
<td>7.12</td>
<td>Team collaboration</td>
<td>cccliii</td>
</tr>
<tr>
<td>7.12.1</td>
<td>Concurrent Versions System (CVS)</td>
<td>cccliii</td>
</tr>
<tr>
<td>7.12.2</td>
<td>Rational ClearCase LT</td>
<td>cccliii</td>
</tr>
</tbody>
</table>
7.12.3 Software configuration and library manager (SCLM) ................ cccliv
7.13 Integrated debuggers .................................................. ccclv
  7.13.1 Java development tools debugger ................................. ccclv
  7.13.2 Server-side JavaScript debug adapter .......................... ccclv
  7.13.3 Active Script debugger ............................................. ccclv
  7.13.4 SQL stored procedure debugger ................................ccclv
  7.13.5 SQLJ debugger ..................................................... ccclv
  7.13.6 Compiled language debugger ..................................... ccclvi
7.14 Profiling and logging tools ........................................... ccclvi
  7.14.1 Log Analyzer ...................................................... ccclvi
  7.14.2 Profiling tool ..................................................... ccclix
7.15 Component test tools .................................................. ccclxii
7.16 Server tools for testing and deployment ............................. ccclxiii
  7.16.1 WebSphere test environment ..................................... ccclxv
  7.16.2 Application publishing .......................................... ccclxvii
  7.16.3 Universal Test Client ............................................. ccclxvii
  7.16.4 Server administration from WebSphere Studio ................ccclxviii
7.17 Business integration tools ........................................... ccclxix
  7.17.1 Services-oriented architecture .................................. ccclxx
  7.17.2 Business Integration perspective .............................. ccclxx
  7.17.3 Human Interaction support ...................................... ccclxx
  7.17.4 Event triggering .................................................. ccclxxi
  7.17.5 Compensation support ............................................ ccclxxi
  7.17.6 Process Debugger ................................................ ccclxxi
  7.17.7 Fault handling .................................................... ccclxxi
  7.17.8 Extended messaging support .................................... ccclxxii
  7.17.9 Application integration ......................................... ccclxxii
7.18 z/OS support tools ..................................................... ccclxxii
  7.18.1 z/OS development tools .......................................... ccclxxiii
  7.18.2 Required z/OS server tools ..................................... ccclxxv
7.19 XML enablement (host) tools ......................................... ccclxxvi
  7.19.1 Inputs ............................................................ ccclxxvi
  7.19.2 Outputs .......................................................... ccclxxvi
  7.19.3 Restrictions ...................................................... ccclxxvii
7.20 WebSphere Studio Client for iSeries tools .......................... ccclxxvii
  7.20.1 WebFacing tool ................................................... ccclxxviii
  7.20.2 Web tools - iSeries extensions .................................. ccclxxviii
  7.20.3 Java tools for iSeries .......................................... ccclxxix
  7.20.4 Remote System Explorer (RSE) perspective .................... ccclxxx
  7.20.5 iSeries Projects perspective ................................... ccclxxx

Chapter 8. IBM WebSphere Commerce ........................................ 1
8.1 Runtime architecture .................................................... 2
8.1.1 WebSphere Commerce software components .................................. 4
8.1.2 WebSphere Commerce Server subsystems ....................................... 8
8.1.3 WebSphere Commerce administration tools .................................. 14
8.2 Business and store models ............................................................... 23
8.2.1 Business and store model overview ............................................. 23
8.2.2 Business model infrastructure and architecture ............................ 30
8.2.3 Store architecture ......................................................................... 39
8.3 Programming model .......................................................................... 59
8.3.1 WebSphere Commerce Server framework ................................... 61
8.3.2 Application flow of an HTTP request .......................................... 68
8.3.3 Customizing application assets ..................................................... 71

Chapter 9. IBM WebSphere Portal V5.0 ..................................................... 97
9.1 WebSphere Portal architecture ......................................................... 98
9.2 WebSphere Portal product editions .................................................. 98
9.3 Portlets ............................................................................................ 102
  9.3.1 Portlet applications ...................................................................... 104
  9.3.2 Portlet modes ................................................................................ 105
  9.3.3 Portlet API .................................................................................. 105
  9.3.4 Portlet performance ...................................................................... 106
  9.3.5 Standards ................................................................................... 106
  9.3.6 Struts ......................................................................................... 107
  9.3.7 Portlet cooperation ...................................................................... 107
  9.3.8 Brokered cooperation .................................................................. 108
  9.3.9 Discoverable services ................................................................... 110
  9.3.10 Tools ....................................................................................... 110
9.4 Content and search ........................................................................... 111
  9.4.1 Syndicated content ....................................................................... 111
  9.4.2 Web content management ............................................................ 112
  9.4.3 WebSphere Portal Content Publishing ....................................... 113
  9.4.4 Workflow .................................................................................... 114
  9.4.5 Enterprise content management .................................................. 115
  9.4.6 Document management ............................................................... 115
  9.4.7 Search ....................................................................................... 117
  9.4.8 Federated search .......................................................................... 119
  9.4.9 Extended search .......................................................................... 119
  9.4.10 Advanced search and document processing ............................. 120
9.5 Security ............................................................................................. 121
  9.5.1 Member services .......................................................................... 121
  9.5.2 Administration ............................................................................. 123
  9.5.3 Authentication .............................................................................. 124
  9.5.4 Identifying the user ....................................................................... 124
  9.5.5 Third-party authentication servers .............................................. 124
Help from IBM ................................................................. 168

Index ................................................................. 169
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

© Copyright IBM Corp. 2004, 2005. All rights reserved.
### Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

<table>
<thead>
<tr>
<th>@server®</th>
<th>ETE®</th>
<th>RAA®</th>
</tr>
</thead>
<tbody>
<tr>
<td>@server®</td>
<td>Informix®</td>
<td>RACF®</td>
</tr>
<tr>
<td>Eserver® (black-e)</td>
<td>Integrated Language Environment®</td>
<td>S/390®</td>
</tr>
<tr>
<td>Eserver® (red-e)</td>
<td>Intelligent Miner®</td>
<td>Sametime®</td>
</tr>
<tr>
<td>Redbooks (logo)®</td>
<td>IBM®</td>
<td>Sequent®</td>
</tr>
<tr>
<td>“Eserver”</td>
<td>IMS®</td>
<td>SupportPac®</td>
</tr>
<tr>
<td>developerWorks®</td>
<td>Language Environment®</td>
<td>SAA®</td>
</tr>
<tr>
<td>e-business on demand®</td>
<td>Lotus Discovery Server®</td>
<td>Tivoli®</td>
</tr>
<tr>
<td>ibm.com®</td>
<td>Lotus®</td>
<td>VisualAge®</td>
</tr>
<tr>
<td>iNotes®</td>
<td>Lotus Notes®</td>
<td>Workplace®</td>
</tr>
<tr>
<td>iSeries®</td>
<td>MVS®</td>
<td>XDE®</td>
</tr>
<tr>
<td>z/OS®</td>
<td>MQSeries®</td>
<td>Java®</td>
</tr>
<tr>
<td>zSeries®</td>
<td>Perform®</td>
<td>Windows®</td>
</tr>
<tr>
<td>AIX®</td>
<td>QuickPlace®</td>
<td>Windows NT®</td>
</tr>
<tr>
<td>Balance®</td>
<td>Rational Rose®</td>
<td>Intel®</td>
</tr>
<tr>
<td>ClearCase®</td>
<td>Rational®</td>
<td>MMX®</td>
</tr>
<tr>
<td>Cloudscape®</td>
<td>Redbooks®</td>
<td>Pentium®</td>
</tr>
<tr>
<td>CICS®</td>
<td>Redbooks (logo)®</td>
<td>UNIX®</td>
</tr>
<tr>
<td>Domino.Doc®</td>
<td></td>
<td>Linux®</td>
</tr>
<tr>
<td>Domino®</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2 Universal Database®</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB2®</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Everyplace®</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following terms are trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.
Preface

This collection of papers is intended to help you understand the products that make up the WebSphere family. The first paper gives you a big picture overview of the WebSphere product family and how the products interrelate. Following are papers that discuss individual product or product families in more detail.

This collection will be useful to IBMers, customers, and business partners that are researching solutions to business problems. It is aimed at decision makers and architects.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Carla Sadtler is a WebSphere Specialist at the International Technical Support Organization, Raleigh Center. She writes extensively in the WebSphere and Patterns for e-business areas. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative. She holds a degree in mathematics from the University of North Carolina at Greensboro.

Gennaro Cuomo (Jerry) holds the prestigious title of IBM Distinguished Engineer. Jerry is entering his 18th year at IBM, splitting his time between IBM TJ Watson Research in NY and in IBM Software Group in Research Triangle Park, NC. Jerry is one of the original engineers of the WebSphere Application Server, focusing on infusing the server with innovative technologies that have contributed to WebSphere's market leadership position in the industry. Currently, Jerry is the WebSphere CTO and Director of the WebSphere Technology Institute, whose mission is to define and develop the next generation of middleware technologies for IBM.

John Ganci is a Senior Software Engineer, WebSphere Specialist at the IBM ITSO, Raleigh Center. He writes extensively and teaches classes on WebSphere and related topics. John has 14 years of experience in product and application design, development, system testing, and consulting. His areas of expertise include e-commerce, WebSphere Application Server, pervasive computing, Linux®, and Java™ programming.
Marc Haberkorn joined IBM in June of 2003 after completing a Bachelor of Science degree in Computer Engineering at Penn State University. Since then, Marc has worked in the IBM WebSphere Technology Institute on various projects, including WebSphere XD. On that project, Marc was a member of the Visualization team, and also contributed in the areas of back end infrastructure code and architecture.

Carol Jones is a Distinguished Engineer in the United States. She has worked at IBM for 16 years, leading various software projects related to Web application technology. She holds a Masters degree in Computer Studies from North Carolina State University. She has written numerous technical publications in the areas of portals, multimedia, Java, and Web application development.

Peter Kovari is a WebSphere Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively about all areas of WebSphere. His areas of expertise include e-business, e-commerce, security, Internet technologies and mobile computing. Before joining the ITSO, he worked as an IT Specialist for IBM in Hungary.

Kevin Griffith is an IBM Certified IT Specialist in France. He has 12 years of experience in IBM application development tools. He is a graduate of the United States Military Academy at West Point and holds a masters degree in Civil Engineering from the Massachusetts Institute of Technology. His areas of expertise include model-driven development and rapid application development tools.

Dildar Marhas is a Senior IT Specialist in IBM Global Services, USA. He has 15 years of software development experience in client-server and distributed systems. He holds a Master’s degree in electrical engineering from Indian Institute of Technology, Powai, India. His areas of expertise include J2EE software development and application integration.

Rob Will is a Distinguished Engineer in the United States. He has worked at IBM for 22 years and on WebSphere since Version 1. He holds a Masters degree in Mathematics from Auburn University.

Thanks to the following people for their contributions to this project:

Suzanne Dewitt
Product Manager, WebSphere Extended Deployment
Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online Contact us review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HZ8  Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195
Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-6963-02
for WebSphere Product Family Overview and Architecture
as created or updated on May 5, 2006.

February 2005, Third Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information
► WebSphere Application Server V6
► WebSphere Extended Deployment V5.1
► Rational Application Developer
► Rational Web Developer
IBM WebSphere product overview

IBM WebSphere is the leading software platform for e-business on demand. Providing comprehensive e-business leadership, WebSphere is evolving to meet the demands of companies faced with challenging business requirements, such as the need for increasing operational efficiencies, strengthening client loyalty, and integrating disparate systems. WebSphere provides answers in today's challenging business environments.

IBM WebSphere is architected to enable you to build business-critical applications for the Web. WebSphere includes a wide range of products that help you develop and serve Web applications. They are designed to make it easier for clients to build, deploy, and manage dynamic Web sites more productively.

In this paper we take a high-level look at the following WebSphere and supporting products:

- IBM WebSphere Application Servers
- IBM WebSphere Studio
- IBM Rational Developer
- IBM WebSphere Business Integrators
- IBM WebSphere Commerce
- IBM WebSphere Portal
1.1 WebSphere overview

WebSphere is the IBM brand of software products that are designed to work together to help deliver dynamic e-business quickly. WebSphere provides solutions for having a positive effect on a client’s business. It also provides solutions for connecting people, systems, and applications with internal and external resources. WebSphere is based on infrastructure software (middleware) designed for dynamic e-business. It delivers a proven, secure, and reliable software portfolio that can provide an excellent return on investment.

The technology that powers WebSphere products is Java. Over the past several years, many software vendors have collaborated on a set of server-side application programming technologies that help build Web accessible, distributed, platform neutral applications. These technologies are collectively branded as the Java 2 Platform, Enterprise Edition (J2EE) platform. This contrasts with the Java 2 Standard Edition (J2SE) platform, with which most clients are familiar. J2SE supports the development of client-side applications with rich graphical user interfaces (GUIs). The J2EE platform is built on top of the J2SE platform. J2EE consists of application technologies for defining business logic and accessing enterprise resources such as databases, Enterprise Resource Planning (ERP) systems, messaging systems, e-mail servers, and so forth.

The potential value of J2EE to the client is tremendous. Among the benefits of J2EE are the following:

- Promotion of an architecture-driven approach to application development helps reduce maintenance costs and allows for construction of an Information Technology (IT) infrastructure that can grow to accommodate new services.

- Application development is focused on unique business requirements and rules, rather than common application aspects, such as security and transaction support. This improves productivity and shortens development cycles.

- Industry standard technologies allow clients to choose among platforms, development tools, and middleware to power their applications.

- Embedded support for Internet and Web technologies allows for a new breed of applications that can bring services and content to a wider range of customers, suppliers, and others, without creating the need for proprietary integration.

Another exciting opportunity for IT is Web services. Quite simply, Web services allow for the definition of functions or services within an enterprise that can be accessed using industry standard protocols that most businesses already use today, such as HTTP and XML. This allows for easy integration of both intra- and
inter-business applications that can lead to increased productivity, expense reduction, and quicker time to market.

1.2 WebSphere family

The WebSphere platform forms the foundation of a comprehensive business solutions framework. Its extensive offerings are designed to solve the problems of companies of all different sizes. For example, the technologies and tools at the heart of the WebSphere platform can be used to build and deploy the core of an international financial trading application. Yet, it also fits very nicely as the Web site solution for a neighborhood restaurant that simply wants to publish an online menu, hours of operation, and perhaps provide a Web-based table reservation or food delivery system. WebSphere's complete and versatile nature can sometimes be the source of confusion for people who are trying to make important decisions about platforms and developer toolkits for their business or departmental projects. So, the goal of this paper is to help you get started with understanding the technologies, tools, and offerings of the WebSphere platform.

Figure 1-1 on page xxvi shows a high-level overview of the WebSphere platform.
1.3 IBM WebSphere Application Servers

WebSphere Application Servers are a suite of servers that implement the J2EE specification. This simply means that any Web applications that are written to the J2EE specification can be installed and deployed on any of the servers in the WebSphere Application Server family.

The primary component of the WebSphere Application Server products is the application server, which provides the environment to run your Web-enabled
Chapter 1. IBM WebSphere product overview

e-business applications. You can think of an application server as “Web middleware,” the middle tier in a three-tier e-business environment. The first tier is the Web server that handles requests from the browser client. The third tier is the business database, for example DB2 UDB, and the business logic, for example, traditional business applications such as order processing. The middle tier is IBM WebSphere Application Server, which provides a framework for consistent, architected linkage between the HTTP requests and the business data and logic.

WebSphere Application Servers are available in multiple packages to meet specific business needs. They also serve as the base for other WebSphere products, such as WebSphere Commerce, by providing the application server required for running these specialized applications.

WebSphere Application Servers are available on a wide range of platforms, including UNIX®-based platforms, Microsoft® operating systems, IBM z/OS, and IBM eServer® iSeries.

Figure 1-2  WebSphere Application Server product overview
1.3.1 WebSphere Application Server V6 for distributed platforms

The latest product to be announced in the WebSphere Application Server family is IBM WebSphere Application Server V6. It features:

- Full J2EE 1.4 support
- High-performance connectors to many common back-end systems, reducing the coding effort required to link dynamic Web pages to real line-of-business data.
- Application services for session and state management
- Web services
  Web services enable businesses to connect applications to other business applications, to deliver business functions to a broader set of clients and partners, to interact with marketplaces more efficiently, and to create new business models dynamically.
- A fully integrated JMS 1.1 messaging provider
  This messaging provider complements and extends WebSphere MQ and application server. It is suitable for messaging among application servers and for providing messaging capability between WebSphere Application Server and an existing WebSphere MQ backbone.
- Many of the programming model extensions previously found in WebSphere Business Integration Server Foundation

Because varying e-business application scenarios require different levels of application server capabilities, WebSphere Application Server is available in multiple packaging options. Although they share a common foundation, each provides unique benefits to meet the needs of applications and the infrastructure that supports them. At least one WebSphere Application Server product package fulfills the requirements of any particular project and the prerequisites of the infrastructure that supports it. As your business grows, the WebSphere Application Server family provides a migration path to higher configurations.

WebSphere Application Server - Express V6

The Express package is geared to those who need to get started quickly with e-business. It is specifically targeted at medium-sized businesses or departments of a large corporation, and is focused on providing ease of use and ease of application development. It contains full J2EE 1.4 support but is limited to a single-server environment.

WebSphere Application Server - Express is unique from the other packages in that it is bundled with an application development tool. Although there are WebSphere Studio and Rational Developer products designed to support each
WebSphere Application Server package, normally they are ordered independent of the server. WebSphere Application Server - Express includes the Rational Web Developer application development tool. It provides a development environment geared toward Web developers and includes support for most J2EE 1.4 features with the exception of Enterprise JavaBeans (EJB) and J2EE Connector Architecture (JCA) development environments. However, keep in mind that WebSphere Application Server - Express V6 does contain full support for EJB and JCA, so you can deploy applications that use these technologies.

**WebSphere Application Server V6**

The WebSphere Application Server package is the next level of server infrastructure in the WebSphere Application Server family. Though the WebSphere Application Server is functionally equivalent to that shipped with Express, this package differs slightly in packaging and licensing. The development tool included is a trial version of Rational Application Developer, full J2EE 1.4 compliant development tool.

To avoid confusion with the Express package in this document, we refer to this as the base package.

**WebSphere Application Server Network Deployment V6**

WebSphere Application Server Network Deployment is an even higher level of server infrastructure in the WebSphere Application Server family. It extends the WebSphere Application Server base package to include clustering capabilities, Edge components, and high availability for distributed configurations. These features become more important at larger enterprises, where applications tend to service a larger customer base, and more elaborate performance and availability requirements are in place.

Application servers in a cluster can reside on the same or multiple machines. A Web server plug-in installed in the Web server can distribute work among clustered application servers. In turn, Web containers running servlets and Java Server Pages (JSPs) can distribute requests for EJBs among EJB containers in a cluster.

The addition of Edge components provides high performance and high availability features. For example:

- The Caching Proxy intercepts data requests from a client, retrieves the requested information from the application servers, and delivers that content back to the client. It stores cachable content in a local cache before delivering it to the client. Subsequent requests for the same content are served from the local cache, which is much faster and reduces the network and application server load.
- The Load Balancer provides horizontal scalability by dispatching HTTP requests among several, identically configured Web server or application server nodes.

Table 1-1 shows the features included with each WebSphere Application Server packaging option.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Express V6(^1)</th>
<th>base V6</th>
<th>ND V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Application Server</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Deployment manager</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IBM HTTP Server V6 Web server plug-ins</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IBM HTTP Server</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Application Client (not on zLinux)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Application Server Toolkit</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DataDirect Technologies JDBC Drivers for WebSphere Application Server</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Development tools</td>
<td>Rational Web Developer (single use license)</td>
<td>Rational Application Developer Trial</td>
<td>Rational Application Developer Trial</td>
</tr>
<tr>
<td>Database</td>
<td>IBM DB2 Universal Database Express V8.2</td>
<td>IBM DB2 Universal Database Express V8.2 (development use only)</td>
<td>IBM DB2 UDB Enterprise Server Edition V8.2 for WebSphere Application Server Network Deployment</td>
</tr>
<tr>
<td>Production ready applications</td>
<td>IBM Business Solutions</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Tivoli Directory Server for WebSphere Application Server (LDAP server)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### 1.3.2 WebSphere Application Server V5.1

WebSphere Application Server V5.1 provides full support for J2EE 1.3 applications. WebSphere Application Server is available in five different configurations:

- WebSphere Application Server - Express
- WebSphere Application Server
- WebSphere Application Server Network Deployment
- WebSphere Application Server for z/OS
- WebSphere Application Server for iSeries

#### WebSphere Application Server - Express V5.1

The Express configuration is a low-cost, easy-to-use, out-of-the-box solution that supports simple, dynamic Web sites based on the Java servlet, JSPs, and Web Services technologies. Express allows for the rapid deployment of Web applications while requiring near zero maintenance.

The Express configuration is specifically targeted at medium-sized businesses or departments of a large corporation, and is focused on providing ease of use and ease of application development. It provides near zero administration overhead, one button install, and comes bundled with an effective development tool.

WebSphere Application Server - Express V5.1 is an extremely capable application server that nearly implements the full J2EE specification. The only J2EE specification left out of WebSphere application Server - Express V5.1 is Enterprise JavaBeans (EJB).
WebSphere Application Server - Express V5.1 also contains enterprise-class connection frameworks that allow developers simple hooks into relational database systems. These connection frameworks are the same modules that ship with more advanced versions of the application server.

WebSphere Application Server - Express V5.1 is bundled with WebSphere Studio Site Developer, an application development tool uniquely suited for the Express offering.

**WebSphere Application Server V5.1**
The WebSphere Application Server base configuration is the next level of server infrastructure in the WebSphere Application Server family.

This configuration is for clients that want to use the full range of J2EE V1.3 technologies, including EJBs and Java Message Service (JMS), but do not need workload management or central administration capabilities.

**WebSphere Application Server Network Deployment V5.1**
WebSphere Application Server Network Deployment extends the base configuration to include clustering capabilities, edge services, and high availability for distributed configurations. These features become more important at larger enterprises, where applications service a larger customer base, and more elaborate performance and availability requirements are in place.

Application servers in a cluster can reside on the same or multiple machines. A Web server plug-in installed in the Web server can distribute work among clustered application servers. In turn, Web containers running servlets and JSPs can distribute requests for EJBs among EJB containers in a cluster.

The addition of Edge Components provides high performance and high availability features.

**WebSphere Application Server for z/OS V5.1**
WebSphere Application Server for z/OS is specifically tailored for the z/OS platform and designed to take advantage of the IBM Eserver zSeries and z/OS qualities of service. From a programming model standpoint, it has the equivalent function found in WebSphere Application Server and WebSphere Application Server Network Deployment.
Numerous run-time topologies are possible, ranging from a simple base configuration with one or several independent application servers up to a complex network configuration running on multiple cells defined across multiple systems in a sysplex.

For information about WebSphere Application Server for z/OS, see:

http://www.ibm.com/software/webservers/appserv/zos_os390/

**WebSphere Application Server for iSeries V5.1**

Although branded for iSeries, the WebSphere Application Server products for iSeries are functionally equivalent to those for the UNIX and Microsoft platforms.

For information about WebSphere Application Server for iSeries, visit:

- WebSphere Application Server for iSeries
- WebSphere Application Server Network Deployment for iSeries
- WebSphere Application Server - Express for iSeries
- WebSphere Application Server for iSeries (Developers edition)

### 1.3.3 WebSphere Extended Deployment V5.1

WebSphere Extended Deployment is designed to take WebSphere Application Server Network Deployment to a higher level. Suited for large and complex runtime environments, WebSphere Extended Deployment offers significant performance, management, and availability features.

For further information, visit the home Web page for WebSphere Extended Deployment:


**On demand operating environment**

WebSphere Extended Deployment allows your application environment to scale on demand with virtualization of WebSphere resources and use of a
goals-directed infrastructure, helping you increase the speed at which your company can adapt to business change.

With a virtualized environment, WebSphere resources are pooled and shared among applications. Instead of deploying applications directly onto a server, you can map an application into a resource pool. This application can then be deployed on some subset of servers within that pool according to your configured business goals.

With virtualization, resources are only allocated to applications as needed. In the event of workload spikes, the resources can be allocated where they are needed most. This efficiency in resource utilization results in better usage of computing resources, allowing you to get the maximum throughput possible on existing machines.

Dynamic clustering provides for autonomic expansion and contraction of a cluster to accommodate periodic spikes in demand for an application. The strategy for increasing these resources is dictated by operational policies that reflect business goals.

The goals-directed infrastructure capabilities of WebSphere Extended Deployment mean that user requests are classified, prioritized, queued and routed to servers based on application and operational policies that are tied to business goals. Application performance is optimized according to policies that reflect service-level goals and relative importance to the organization.

Transactions are grouped into transaction classes, which are then mapped into service policies. Incoming requests are classified and queued according to service policies by an on demand router. Autonomic managers monitor performance statistics and dynamically manage system resources to optimize performance. Statistics regarding the performance are available in customizable views.

Simply put, you can state what applications are important to you, and these applications will get the highest-priority access to your WebSphere resources at the right time. This can help you ensure that your business-critical transactions get the best quality of service.

**Extended manageability**

WebSphere Extended Deployment extends the WebSphere Application Server administrative console to address the special needs of very complex environments. These enhancements include:

- Views of complex, dynamic topologies that provide snapshots of the state of the runtime environment
- Customizable graphs of run-time data that provide performance and throughput information
- Notification of decisions made by autonomic managers

Notifications for planned events can be presented for manual or supervisory action, or can simply indicate that an autonomic manager has reacted to a situation. Notifications for unplanned events are displayed to let the administrator know that an event has happened that may need corrective action.

**High-performance computing**

WebSphere Extended Deployment enhances quality of service of business-critical applications to support near linear scalability for high-end transaction processing, helping you improve customer service levels while also leveraging existing Java skills and resources. Dynamic application partitioning using the WebSphere partition facility provides the foundation for the high-performance features. Workload management and high-end caching are both enhanced through the use of partitions.

### 1.4 IBM software development platform

The IBM software development platform in Figure 1-3 on page xxxvi consists of a set of integrated tools, best practices, and services that support a proven end-to-end process for the application development life cycle.
1.4.1 Application development for WebSphere Application Server V6

With V6 of WebSphere Application Server, the WebSphere Studio development products supporting it have been rebranded as follows:

- Rational Web Developer is the V6 follow-on to WebSphere Studio Site Developer.
- Rational Application Developer is the V6 follow-on to WebSphere Studio Application Developer.

Based on Eclipse V3, these development products have the same look and feel as the WebSphere Studio products. The focus of the updates to these products is centered on:

- Ease of use
- Rapid application development and deployment
- Simplified application development and service construction
- Integrated process guidance and tool assistance

Note: Rational Performance Analyst is intended for both Developers and Testers
1.4.2 WebSphere Studio and Rational Developer

The Rational Developer and WebSphere Studio application development tools provide a comprehensive development environment designed to meet all of your development needs:

- state-of-the-art Web interfaces
- server-side applications
- individual development
- advanced team environments
- Java development
- application integration

Available in a number of configurations, with extensions from IBM and IBM Partners, these products enable developers to use a single development environment designed to meet their specific development needs.

Because these products are based on the open-source Eclipse platform, they provide an open, extensible plug-in architecture. Numerous plug-ins are available from IBM Partners and the open source community. Or, using the included plug-in development environment, you can create your own plug-ins.

WebSphere Studio and Rational developer products are designed to be used by a wide variety of roles, such as Web developers, Java developers, enterprise programmers, business analysts, and system architects. From one development environment you can develop, test, deploy, and manage applications. A rich set of utilities and wizards help simplify common tasks so that developers can concentrate on providing true business value and on getting robust applications rapidly into production.

WebSphere Studio and Rational development products include productivity enhancing tools such as:

- A tool for visually managing a Web site
- Wizards generating specification-compliant artifacts
- Automatically updating Web links as you move resources around
- Rich media tools enhancing images and creating JavaScript to add visual interest to Web sites
- Built-in refactoring tools making it easy to reorganize Java classes
- An integrated unit test environment with hot deployment capabilities
- Profiling tools for detecting and analyzing performance problems

In addition to providing plug-ins that support Concurrent Versions Systems (CVS) and Rational ClearCase LT, the open Eclipse framework allows you to add
adapters available from the open source community, IBM, and other commercial vendors.

Note: To obtain more about how WebSphere Studio and Rational Developer can speed up the development process see the IBM White Paper: *Rapid Java and J2EE development with IBM WebSphere Studio and IBM Rational Developer* at:


Figure 1-4 on page xxxix shows the user roles, main functions, and target runtimes of the Rational Developer and WebSphere Studio products targeted to the application developer. The base user interface and environment functions are provided by the Eclipse platform, along with frameworks to allow tool builders to develop plug-ins.

Each configuration includes a common workbench and a set of additional plug-ins that provide development capability customized for application development to a specific runtime environment.
All WebSphere Studio and Rational Developer products are designed to run in a Windows® or Linux environment.

**Rational Web Developer**
Rational Web Developer, formerly known as WebSphere Studio Site Developer, is designed for the entire Web development team:

- Content authors
- Graphic artists
- Web programmers
- Web masters

Rational Web Developer makes it easy to collaboratively design, create, assemble, publish, deploy, and maintain dynamic, interactive Web applications that exploit today’s open technology standards.

---

**Figure 1-4  Software application development tools**

All WebSphere Studio and Rational Developer products are designed to run in a Windows® or Linux environment.

**Rational Web Developer**
Rational Web Developer, formerly known as WebSphere Studio Site Developer, is designed for the entire Web development team:

- Content authors
- Graphic artists
- Web programmers
- Web masters

Rational Web Developer makes it easy to collaboratively design, create, assemble, publish, deploy, and maintain dynamic, interactive Web applications that exploit today’s open technology standards.
You can quickly build and test business logic, and enhance presentation artifacts with the built-in Web creation tools before deploying on a production server such as WebSphere Application Server or Apache Tomcat.

This configuration includes an intuitive WYSIWYG editor that lets novice Web designers create and publish Web sites while incorporating the latest Web technology, including JavaScript, dynamic HTML, and Cascading Style Sheets.

Visual layout tools are provided to create dynamic Web sites with Java servlet or JSP components. Included is a built-in XML development environment, along with Web services tools for the creation, deployment, and publishing of Web services.

Struts tools assist in the creation of well-architected Web applications, while support for JavaServer Faces simplifies the construction of Web interfaces and Web applications. Procedural programmers will find Java application development easy with the Enterprise Generation Language (EGL) tools.

Rational Web Developer also provides an interface to Concurrent Versions System (CVS) for team development and version control.

Rational Web Developer is packaged with WebSphere Application Server - Express V6. For more information, visit the home Web page for Rational Web Developer:

http://www.ibm.com/software/awdtools/developer/web

**Rational Application Developer**

Rational Application Developer includes all of the features of Rational Web Developer plus additional features, including:

- EJB development tools
- Performance profiling and logging tools for both local and remote execution
  
  With these tools, it is possible to detect application performance bottlenecks early in the development cycle.
- UML Visual Editor for Java and EJB
- CVS interface tools plus Rational ClearCase LT for version control
- The IBM Portal Toolkit,

  The Toolkit enables developers to create, edit, and test portlets that comply with the JSR 168 architecture and with the IBM Portlet API (an extension of JSR 168). Portlets can be created using the Struts framework and can include JavaServer Faces UI components.
The target runtime environment is a J2EE server, such as the WebSphere Application Server base and Network Deployment configurations. The built-in test environment for WebSphere Application Server and advanced tools for code generation help shorten the test cycle.

For further information, visit the Web home page for Rational Application Developer:

http://www.ibm.com/software/awdtools/developer/application

**WebSphere Studio Application Developer Integration Edition**

WebSphere Studio Application Developer Integration Edition includes the traditional J2EE development and test features found in WebSphere Studio tools. In addition, it provides tools that focus on integration with back-end systems:

- **Enterprise services**

  Enterprise services is a set of tools and wizards that facilitate service-oriented development. Included are tools to provide support for consuming services from and providing services to WebSphere Business Integration Server Foundation, which acts as the point of integration for a wide variety of services.

  Various resources are consumable, such as SOAP Web services, Java beans, stateless session (EJB) beans, and JCA services. Services deployed into the WebSphere Application Server can be provided as SOAP services through the EJB programming model.

  Enterprise services are based on open standards such as J2EE, JCA, Web Services Description Language (WSDL), Web Services Invocation Framework (WSIF), and Extensible Stylesheet Language Transforms (XSLT).

- **Graphical business process composition tools**

  These tools enable you to visually compose a service out of one or more existing services, define and transform the flow of information between services, and create a business process that contains other nested processes.

- **Development connectors**

  These connectors are for integration with back-end systems, Application Developer Integration Edition provides development connectors for Customer Information Control System (CICS), Information Management System (IMS) and Host-on-Demand (HOD). IBM provides a range of adapters for popular packaged applications including Customer Relationship Management, Supply Chain Management, Enterprise Resource Planning and mainframe applications.
The target application server for Application Developer Integration Edition is WebSphere Business Integration Server Foundation.

For further information, visit the Web home page for WebSphere Studio Application Developer Integration Edition:

http://www.ibm.com/software/integration/wsadie/

WebSphere Studio Enterprise Developer
Enterprise Developer provides enterprise-level development tools for interacting with back-end development and operating systems. In addition to many of the Application Developer Integration Edition features, Enterprise Developer provides the following:

- z/OS application development tools in an interactive, workstation-based environment for performing the following tasks:
  - Perform MVS-based system tasks such as managing datasets with a workstation-like directory structure and submitting JCL or TSO commands.
  - Create HLASM, COBOL, or PL/I code, including CICS, IMS, and SQL statements.
  - Edit, test, and compile the source code locally or remotely on a z/OS system.
  - Transfer CLISTs and REXX EXECs scripts to z/OS, run them, and view the resulting output.

- Enterprise Generation Language (EGL) tools
  EGL tools provide a common language for implementing business processes targeting a variety of runtime platforms. EGL lets traditional enterprise developers rapidly deliver enterprise-wide data to browsers, regardless of the depth of their Web or object-oriented development expertise. Enterprise Developer provides EGL code generation support (via a value add plugin) that produces COBOL code.

- Integration with existing transactional environments such as CICS and IMS

- Tools for creating new Enterprise Information System (EIS) applications as part of a J2EE application

- Integration of Struts-based Model-View-Controller applications using J2EE Connector Architecture connectors and EGL

- Integration with WebSphere Studio Asset Analyzer (WSAA) to identify application processes and connecting points and provide the ability to generate components from existing code
The target runtime environment is WebSphere Application Server Enterprise or WebSphere Application Server for z/OS.

For further information, visit the Web home page for WebSphere Studio Enterprise Developer:

http://www.ibm.com/software/awdtools/studioenterprisedev/

**WebSphere Development Studio for iSeries**

WebSphere Development Studio for iSeries provides tools to increase the number of Web-enabled, e-business applications for the IBM @server iSeries server.

This product includes server and workstation components. The server component, WebSphere Development Studio, includes ILE RPG (RPG IV), ILE COBOL, ILE C/C++, and Application Development ToolSet (ADTS).

The client component, WebSphere Development Studio Client, is an Eclipse-based development environment. WebSphere Development Studio Client for iSeries extends IBM strategic development tools and adds iSeries-specific value. It includes the following development tools:

- IBM WebFacing Tool for Web enabling 5250-based applications
- Remote System Explorer (RSE), which provides integrated, workstation-based iSeries development functions
- Web tools
- Java tools
- Web Services tools
- Database tools
- XML tools
- CODE (classic version)
- VisualAge RPG (classic version)
- Distributed Debugger (classic version)

For further information, visit the Web home page for WebSphere Development Studio for iSeries:

http://www.ibm.com/software/awdtools/wds400/

### 1.5 IBM WebSphere Business Integration products

Business integration products optimize operations by integrating and automating business processes. WebSphere Business Integration products help clients to interconnect their islands of information and make full use of the message-based architecture.
Historically, isolated applications were written to solve specific problems. There was little vision at the time of an application landscape that would cover the whole range of business requirements, so the need for an integrated architecture wasn’t apparent. As a result, solutions evolved on a variety of platforms. If and where integration was needed, it was usually achieved by hosting the applications on the same system and sharing files. This was no great restriction, since most applications back then were batch-oriented, and large central computers (the “mainframes”) were the accepted technology standard.

Where online processing became possible, businesses found it more acceptable from a risk and system capacity perspective to just collect data during the day (in files) and do the actual processing during nightly batch runs. This mode of operation is still quite prevalent in businesses today.

With the evolution of database management systems, the models surrounding the sharing of information began a trend towards integrating information across systems via database federation, transformation, and replication technologies. It required the systems sharing information to share the physical data models, processing rules, and constraints. This sharing resulted in tight coupling between applications. Still, in the real world of IT, which includes legacy technology and closed systems, there are many cases where this is the only feasible style of integration, and in those cases it provides good service. Incidentally, this integration approach coincided with 2-tier client-server architectures and continues to be in use.

Integration at a fundamental level is about making sure information from different business applications can be shared and used across and beyond the enterprise in a reliable and secure manner through application connectivity. With the evolution of distributed object and message-oriented middleware technologies, it is now possible to integrate isolated applications running on heterogeneous platforms. This approach allows legacy applications to be connected together with minimal changes needed in existing code. Applications continue to use their own data formats while an “integration broker” provides the necessary mapping, augmentation, and distribution of data at runtime. Message-oriented middleware allows event-based loose coupling between the connected applications and offers increased reliability and flexibility.

Process integration is at the highest level of business integration. It implies coordinating and controlling activities that may span multiple systems and involve people in a variety of roles. It structures, implements, automates, and manages business processes while providing runtime measurements that will then assist in optimizing the process models. Process integration can support long-running transactions and roles-based human activities. The flow of a business event through the process can be modified by external input, either by parameters provided when the process is instantiated or by information retrieved from
external data sources such as an application database, or by human decisions such as in an approval step.

Process integration can also be seen as the business logic layer that determines what needs to be done at a given point in a process, as opposed to how it gets done, which is typically the role of the application. Separating the what from the how allows flexibility since one can be changed without affecting the other.

1.5.1 Integration servers

Integration servers provide the glue, or bond, between existing applications. Applications are connected to the integration server by using a variety of adapters and plug-ins. These applications can then communicate with each other in one of the following ways:

- Sending and receiving messages
- Event notifications mediated by the adapters
- Using service access calls based on IIOP for CORBA technology, RMI/IIOP for EJB technology, and JCA

The integration servers provide three distinct flavors of runtime environments that execute the appropriate integration processes:

- Message-flows
- Collaborations
- Workflows

Message-flows and collaborations represent two different levels of integration. In either case, applications send and receive data and objects in their native formats. The message-flows and collaborations perform the transformation, augmentation, and routing required for integration.

Collaborations emphasize business processes, business objects, a common business object model, business events, and rules. They are best suited for low throughput and response time requirements. Message-flows, on the other hand, emphasize fast message transformation and large throughput capability. Message-flows represent functional integration. Collaborations represent process integration and form parts of higher-level integrated business processes implemented as workflows.

Message-flows, collaborations, and workflows can be visually modeled, managed, and optimized. Collaborations and workflows can also be visually monitored. This reduces complexity and enhances the ability to adapt systems to constantly changing business needs, thus forming a sound operational foundation for on demand business.
1.5.2 Product overview

The IBM WebSphere Business Integration products cover the whole spectrum of business integration, from bottom-up application connectivity to top-down business process automation. The IBM WebSphere Business Integration product family provides the following capabilities:

- **Model** and simulate the enterprise business processes
- **Integrate** the islands of processing
- **Connect** clients and business partners
- **Monitor** the business process end-to-end
- **Manage** the effectiveness and improvement of business processes

Figure 1-5 shows the WebSphere Business Integration family of products.

![Image](image.png)

**Figure 1-5**  WebSphere Business Integration products and features

1.5.3 WebSphere Business Integration Server Foundation

WebSphere Business Integration Server Foundation V5.1, formerly known as WebSphere Application Server Enterprise, along with WebSphere Studio Application Developer Integration Edition V5.1 for Windows or Linux provides the
foundation for building and deploying service-oriented applications that can adapt quickly and easily to change.

WebSphere Business Integration Server Foundation provides the features found in the WebSphere Application Server Network Deployment V5.1, plus programming model extensions for complex application designs and the Business Process Choreographer for business process applications. It will provide the base for future WebSphere Business Integration solutions.

It offers such capabilities as advanced application adapters, application workflow composition and choreography, extended messaging, dynamic rules-based application adaptability, internationalization, and asynchronous processing. It supports the creation of new services and services based on existing applications and systems, which can then be combined to form composite applications and business processes.

Features of WebSphere Business Integration Server Foundation include:

- **Service-Oriented Architecture (SOA)**
  Applications need to integrate business logic and application data within the organization and outside. SOA represents all software assets as services, including legacy applications, packaged applications, J2EE components, and Web services. It also provides standards to interact between the assets. The individual assets can be reused as building block in other applications.

- **Integrated workflow support**
  Using services in a business process is the next step in application development. The integrated workflow capabilities offer flow-based application development based on Business Process Execution Language for Web Services (BPEL4WS). In these flows, different software assets and services can be reused and composed into a business process.
  The built-in engine supports human interaction and compensation for rollback-like transaction support.

- **Advanced transactional connectivity**
  WebSphere Business Integration Server Foundation provides numerous transaction support features essential for large enterprise applications, including dynamic application adapter support, last participant and ActivitySession service support, and CORBA C++ support.

- **Optimized application performance**
  With application profiling techniques, applications can be fine-tuned for better performance without changing a single line of code. Network Deployment provides scalability, performance, availability, and centralized management for WebSphere Business Integration Server Foundation.
Next generation development

WebSphere Business Integration Server Foundation and WebSphere Studio Application Developer Integration Edition enable “next generation” development by leveraging the latest innovations that build on today's J2EE standards, providing greater control over application development, execution, and performance than was ever possible before. These include asynchronous beans, startup beans, scheduler service, and object pools.

Increased development productivity

One way to vastly improve developer productivity is to reduce the need for handcrafted solutions that can be time-consuming, costly, and difficult to maintain. WebSphere Business Integration Server Foundation and WebSphere Studio Application Developer Integration Edition were designed to improve developer productivity through the use of extended messaging capabilities, internationalization service, and work areas.

Real-time application flexibility

In order for businesses to respond quickly to changes, applications must be able to make changes and adopt to current conditions without changing the application code. Business rule beans provide a real-time framework for defining, executing, and managing business rules that encapsulate business policies. Dynamic query allows the EJB container, using the EJB Query Language, to submit queries that select, sort, join, and perform calculations on application data at runtime.

For further information, visit the Web home page for WebSphere Business Integration Server Foundation:

http://www.ibm.com/software/integration/wbisf/

WebSphere Business Integration Server Foundation for z/OS

WebSphere Business Integration Server Foundation is now available on the z/OS platform with the same functionality and code base found on the distributed platforms. And, as with the distributed platforms, application development tools are provided by WebSphere Studio Application Developer Integration Edition V5.1 for Windows or Linux.

WebSphere Business Integration Server Foundation features

The features that distinguish WebSphere Business Integration Server Foundation from WebSphere Application Server Network Deployment are targeted in three areas, as shown in Figure 1-6 on page xlix.
**Business object model**

The first group of features targets the business object model, that is, the representation of business data by a network of entity EJBs. These features provide extensions to the EJB 2.0 specifications and to the data access and caching functions supported by the base WebSphere Application Server. They include:

- Application profiling, access intent

  The application profiling and access intent features provide a flexible method to fine-tune application performance for EJBs. Different EJBs or even different methods within an EJB can have their own intent to access resources. Optimizing and profiling components based on their access intent increases the performance at run time.

  WebSphere Studio Application Developer Integration Edition V5.1 includes optimization tooling designed to facilitate the task of creating the access intent characteristics of applications. With a click of a button, the tooling will analyze a J2EE application and define an appropriate application profile.

- Dynamic query

  Dynamic query fixes a weakness of Enterprise JavaBeans by enabling the client to run custom queries on EJBs during runtime. Until now, EJB lookups and field mappings were implemented at development time. Modifying the query attributes required further development. With dynamic query the client can assemble EJB query language (EJB QL) statements at runtime and execute them on any EJB. The queries can result in objects, lists of objects, or the results from aggregate functions, SUM, COUNT, for example.
CMP over Anything

Traditionally, WebSphere Application Server has offered two options for handling persistent data. The first option, bean-managed persistence, relied on the application to implement the persistence logic. The second option, container-managed persistence, left the persistence logic to the application server, but the options were limited to using a relational database via JDBC or SQLJ. With WebSphere Business Integration Server Foundation, the container-managed persistence has been expanded to include the capability of persisting data to any backend. This feature, referred to as CMP over Anything allows mapping CMP EJBs to backends beyond the traditional relational database.

The optimization tooling in WebSphere Studio Application Developer Integration Edition V5.1 complements this feature and enhances the traditional container-managed persistence by assisting you in defining application profiles and access intents for optimal performance.

Business process model

This group of features targets the business logic developer.

Process Choreographer, including staff services

Process choreographer provides workflow and process functionality for the application server. It supports long-running and short-running, interruptible and non-interruptible processes.

The processes are developed in WebSphere Studio Application Developer Integration Edition using a visual editor to assemble the flow. The process application is then deployed in the application server's process container.

The runtime environment also provides a browser-based client application for administrative purposes. The default client can be freely customized and extended using the client API.

The process choreographer has an integral service called staff services, that provides advanced staff and user handling functions beyond the base user registry function.

The Business Process Choreographer supports processes created using FDML (used in WebSphere Application Server Enterprise 5.0) and in Business Process Execution Language for Web Services (BPEL4WS) V1.1.

Activity sessions and last participant support

Last participant support is an extension to the original J2EE transaction support for applications. It allows a single-phase resource to participate in a global transaction. Developers can include several 2-phase commit resources and one, and only one, 1-phase commit resource in one unit of work.
The activity sessions extension provides further enhancements to the J2EE session and transaction services by extending session and transaction boundaries. This extension allows coordinating multiple 1-phase resources into a single logical unit of work, extends the EJB life cycle to provide for better EJB performance in long transaction scenarios, and extends the lifetime of local transactions beyond standard boundaries.

- **Business rule beans**

  Business rule beans externalize the business rules in an application. Rules, conditions, and decisions do not have to be hard-coded or custom tailored into applications. Business rule beans provide an API to look up and use the rules and provide an interface within the administration console to manage application rules.

**Next generation applications**

The third group of features provides advanced application enablements to facilitate the development of sophisticated applications. This group of features includes:

- **Extended messaging**

  Extended messaging provides significant function for building messaging-based applications. It enables you to create quickly applications that integrate with other systems through a messaging infrastructure. It offers automated support for inbound and outbound messaging by hiding the complexities of the messaging API.

  This extension provides an API built on top of JMS. In addition, complementary application development aids are provided in WebSphere Studio Application Developer Integration Edition, allowing you to build messaging components for any messaging patterns with minimum effort.

- **Asynchronous beans**

  Asynchronous beans offer performance enhancements for resource-intensive tasks by enabling single tasks to run as multiple tasks. Asynchronous scheduling facilities can also be used to process parallel processing requests in batch mode at a designated time. WebSphere Business Integration Server Foundation provides full support for asynchronous execution and invocation of threads and components within the application server. The application server provides execution and security context for the components, making them an integral part of the application.

- **Startup beans**

  Startup beans allow the automatic execution of business logic when the application server starts or stops. For example, they might be used to pre-fill application-specific caches, initialize application-level connection pools, or perform other application-specific initialization and termination procedures.
Scheduler service

The scheduler service allows you to schedule processes in a timely manner for batch processing or to maximize the utilization of existing computing resources. The scheduler service provides the ability to process workloads using parallel processing, set specific transactions as high priority, and schedule less time-sensitive tasks to process during low traffic off hours. Two action types can be scheduled. A method on a session EJB can be called and a message can be sent to a JMS destination. The scheduler service uses a calendar to schedule tasks. This can either be a calendar provided by the product, or a user-defined calendar.

Object pools

Application performance is essential in runtime. Object pools provide an effective means of improving performance by allowing multiple instances of objects to be reused, reducing the overhead associated with instantiating, initializing, and garbage-collecting the objects. Creating an object pool allows an application to obtain an instance of a Java object and return the instance to the pool when it has finished using it.

Shared Work Area

In the process of developing software applications, the need to pass data between application components is often a fundamental requirement. Shared Work Areas provide a solution to pass and propagate contextual information between application components.

Internationalization (I18N) service

An application that can present information to users according to regional cultural conventions is said to be internationalized. The application can be configured to interact with users from different localities in culturally appropriate ways. The Internationalization (I18N) service gives the ability to develop internationalized applications in WebSphere.

1.5.4 WebSphere Business Integration Server

At the highest level, a business integration system consists of an integration broker and a set of adapters that allow heterogeneous business applications to exchange data through the coordinated transfer of information, in the form of business objects.

IBM WebSphere Business Integration Server is an offering containing the following IBM WebSphere Business Integration products:

- IBM WebSphere InterChange Server
- IBM WebSphere MQ Workflow
- IBM WebSphere Business Integration Message Broker for Multiplatforms
IBM WebSphere Business Integration Toolset
Entitlement to use a specific set of IBM WebSphere Business Integration Adapters for the purposes of integrating the InterChange Server with the Message Broker and Workflow product components.

For further information, visit the WebSphere Business Integration Server home Web page:

http://www.ibm.com/software/integration/wbiserver/

1.5.5 WebSphere InterChange Server

WebSphere InterChange Server is a runtime environment with graphical user interface (GUI) tools to integrate and manage business processes. It provides a Java-based runtime environment for integrated business process implementations called collaborations. It also provides CORBA, EJB, and Java connector-based synchronous access mechanisms to invoke these business processes.

A WebSphere business integration system implemented with WebSphere InterChange Server uses modular components and application-independent business logic. The InterChange Server approach uses a distributed hub-and-spoke infrastructure. It executes business processes such as e-business order fulfillment, returns processing, and inventory management that are distributed across the Internet, across applications on local networks, or both. The system is distributed and flexible, with reusable components and customization features that make it possible to meet site-specific and application-specific needs.

An implementation of WebSphere InterChange Server utilizes the adapters provided by the WebSphere Business Integration Adapters product, together with several types of modular and customizable components, including collaborations, business objects, maps, and data handlers.

Business-process logic resides in InterChange Server collaborations at the hub. Data is exchanged between the hub and the spokes in the form of business objects. Application and technology adapters, which are available in the IBM WebSphere Integration Adapters product, supply connectivity to applications or to Web servers or other programmatic entities at the spokes. The Server Access Interface makes it possible for remote spoke sites that do not implement WebSphere InterChange Server to makes calls over the Internet to a hub site that does.

Collaborations

Collaborations are software modules that contain code and business process logic that drives interactions between applications. A collaboration can be...
simple, consisting of just a few steps, or complex, involving several steps and other collaborations.

Collaborations can be distributed across any number of applications, can handle synchronous and asynchronous service calls, and can support long-lived business processes.

Transactional collaborations provide data consistency for business processes and support nested transactions, compensated rollback, and transaction recovery.

► Business objects

Business objects provide an application and implementation independent way to describe, exchange, and process data entities. A business object definition specifies the types and order of information in each entity that WebSphere InterChange Server handles, and the verbs that it supports.

The InterChange Server repository stores business object definitions. A business object is an instance of the definition, containing actual data. Business objects are created at runtime and not stored in the repository.

► Data mapping

Data mapping is the process of converting business objects from one type to another. Data mapping is required whenever the WebSphere InterChange Server system sends data between a source and a destination that does not exactly share the source's data model.

Unlike custom application integration solutions that map data directly from one application to another, WebSphere InterChange Server collaborations generally use generic business objects between application-specific data models. The generic business object serves as a common application and implementation independent dataset. When applications change in the future, you need only get a new connector and map the new application-specific business object to the generic business object. Collaborations then continue to work as they did previously.

To actually perform data mapping, the system invokes the use of the Map Designer and Relationship Designer mapping tools. These tools allow creation and modification of detailed mapping specifications at design time and execution of data mapping at runtime.

► Server Access Interface

The CORBA-compliant API of Server Access Interface accepts synchronous data transfers from internally networked or external sources. The data is then transformed into business objects that can be manipulated by a collaboration. The Server Access Interface makes it possible to receive calls from external entities (for example, Web browsers at a remote customer site) that do not
come through connector agents, but instead come through Web servlets into the Server Access Interface.

Reuseable components

The Server Access Interface and the adapters both make use of data handlers. In the WebSphere InterChange Server environment, new data handlers can be created from a modular group of base classes called the Data Handler Framework. The WebSphere InterChange Server solution also includes a Protocol Handler Framework. These frameworks make it easier to customize solutions and add connectivity for additional data formats and protocols in the future.

Options for different types of interactions

WebSphere InterChange Server supports two basic types of interactions: publish-and-subscribe interactions or service call interactions. Both types of interaction supply triggers that start the execution of a collaboration’s business processes. The collaboration then uses a third type of interaction, request/response, to complete the exchange of data with the intended destination.

Depending on the adapter, an event can be published to a collaboration either asynchronously or synchronously. In addition, if the long-lived business process feature of the collaboration is enabled, a collaboration can maintain the event in a waiting state, in anticipation of incoming events satisfying predefined matching criteria.

Access request interactions are useful when synchronous communication is important, for example, a customer representative using a Web browser to request inventory status information over the Internet. These types of connectivity can be developed using J2EE EJB and J2EE Connector Architecture (JCA) using the Server Access Interface API.

For further in information, visit the WebSphere InterChange Server home Web page:


1.5.6 WebSphere MQ

WebSphere MQ is IBM’s award-winning middleware for commercial messaging and queuing. It runs on a variety of platforms. The WebSphere MQ products enable programs to communicate with each other across a network of unlike components, such as processors, subsystems, operating systems, and communication protocols. WebSphere MQ allows you to:

- Connect any commercial systems, with over 35 platforms supported in business today
- Tolerate network disruptions ensuring important data is always delivered.
- Integrate disparate islands of automation.
- Provide time-independent communication.
- Assure one-time delivery.
- Support high-volume throughput with customer experience in excess of 250 million messages a day.
- Provide SSL support.

For further information, see the WebSphere MQ home Web page:

http://www.ibm.com/software/integration/wmq/

Previous versions of WebSphere MQ (IBM MQSeries) may be available for other platforms. Information about these products can also be found on the requirements Web site.

### 1.5.7 WebSphere Business Integration Message Broker

WebSphere Business Integration Message Broker extends the messaging capabilities of WebSphere MQ by adding message routing, transformation, and publish/subscribe features. Message Broker provides a runtime environment that executes message-flows. These message-flows consist of a graph of nodes that represent the processing needed for integrating applications. They can be designed to perform a wide variety of functions, including the following:

- Routing of messages to zero or more destinations based on the contents of the message or message header
  Both one-to-many and many-to-one messaging topologies are supported.

- Transformation of messages into different formats so that diverse applications can exchange messages understandable to each.

- Enrichment of the message content en route, by using a database lookup performed by the message broker, for example.

- Storage of information extracted from messages en route to a database using the message broker

- Publication of messages and using topic- or content-based criteria for subscribers to select which messages to receive

- Interaction with other connectivity mechanisms such as MQSeries Everyplace

- Extension of the base function of WebSphere MQ Message Broker with plug-in nodes in Java and C/C++, which can be developed by installations as well as by IBM and ISVs
Processing message content in a number of message domains, including the XML domain which handles self-defining, or generic, XML messages; the Message Repository Manager (MRM) in which handles predefined message sets, and unstructured data (BLOB domain)

WebSphere Business Integration Message Broker also provides the following:

- Scalability options in the form of message-flow instances and execution-groups
- Simplified integration of existing applications with Web services through the transformation and routing of SOAP messages, as well as logging of Web services transactions
- Mediation between Web services and other integration models as both a service requester and a service provider
- Compliance with standards such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP), and Hypertext Transport Protocol (HTTP)
- Integrated WebSphere MQ transports for enterprise, mobile, real-time, multicast, and telemetry end points
- Eclipse-based Message Broker Toolkit for WebSphere Studio
- Standards-based metadata including XML schema and WSDL

**Selecting a broker:** As you can see, WebSphere Business Integration Server provides two different, yet compatible integration brokers: WebSphere InterChange Server and WebSphere Business Integration Message Broker. There are differences between integration systems using these two brokers, and choosing the right broker is an important high-level decision.

WebSphere InterChange Server is a *process integration* engine. Its primary purpose is to choreograph interactions between a number of applications. As such, it needs to retain state information and handle concepts such as compensating transactions and dynamic cross-referencing.

The WebSphere Business Integration Message Broker, on the other hand, provides *application connectivity* services. They generally act as intermediaries between applications, providing fast routing and transformation services.

The two products also work seamlessly with each other. A broker can act as an intermediary to provide application connectivity services between applications and the WebSphere InterChange Server.
1.5.8 WebSphere Business Integration Adapters

WebSphere Business Integration Adapters enable data to move between an application, a technology, or a packaged legacy system and the underlying transport infrastructure. These adapters access, transform, and route data through the integration brokers.

Imagine a wheel. WebSphere Business Integration Adapters can be considered the spokes that connect applications and technologies to the integration broker hubs. They provide application, technology, mainframe, and e-business connections for both process integration and application connectivity requirements.

- **Application adapters** allow interactions with specific applications and are intermediaries between collaborations or message-flows and applications. These adapters transform data from the application into business objects that can be manipulated by the collaborations and message-flows, and transform business objects from the collaborations and message-flows into data that can be received by the specific application.

- **Technology adapters** allow interactions that conform to specific technology standards. For example, the XML connector can be used for sending data from WebSphere InterChange Server collaborations to a Web server, even if that Web server resides beyond a firewall on a network that is not running the connector agent or other IBM WebSphere software. XML, JDBC, JTEXT, and JMS adapters are examples of technology adapters.

- **Mainframe adapters** allow interactions with legacy applications running on mainframes. (For example, the CICS adapter)

- **E-business adapters** provide proven solutions for securely connecting over the firewall to clients' desktops, to trading partners' internal applications, and to online marketplaces and exchanges. (For example, the E-Mail adapter)

WebSphere Business Integration Adapters are built using a common customizable Java-based framework, and can be deployed on a variety of platforms. All the configuration and development tools are available as Eclipse plug-ins.

WebSphere Business Integration Adapters have the following components:

- An adapter that links the applications to the integration broker at run time
Tools with GUI interfaces that help create business object definitions for the applications and configure adapters

An Object Discovery Agent (ODA) that helps create rudimentary business object definitions from an application's data store

The ODA is not included in every adapter.

An Object Discovery Agent Development Kit (ODK) that consists of a set of APIs to develop an ODA

Separately available Adapter Development Kit (ADK) that provides a framework for developing custom adapters

For a list of the adapters and links to information about each, see the home page for the WebSphere Business Integration Adapters at:


1.5.9 WebSphere Business Integration Toolset

WebSphere Business Integration Toolset is a set of tools that provide administrative and development support for WebSphere InterChange Server. They include tools for system management, application connectivity, and business process modeling. These tools are implemented as Eclipse plug-ins and have the following general features:

- Open standards-based
- Unified interface across the full suite of WebSphere Studio tools
- Instant integration with an array of Eclipse plug-ins
- Source code control
- Team development
- XML Schema Definition (XSD) for all WebSphere InterChange Server artifacts
- XMI support for UML import/export of process templates
- Import/export collaboration process using a subset of BPEL
- Expose business processes as Web services
- Invoke external Web services
- Web Services Object Discovery Agent with UDDI support
- Minimize/eliminate need to edit Java code
- Visual support for basic functions: dates, mathematical, strings, and so forth
- Message files in consistent table format, not free form text
- Declarations set via drop-downs, not in code
- Integrated test environment
WebSphere Business Integration Toolset for Administration

The WebSphere Business Integration Toolset for Administration is used to manage and monitor various WebSphere InterChange Server system environments. Some of the capabilities the tools provide are the following:

- **Flow Manager**
  - Graphically queries unresolved flows
  - Provides detailed messages on the cause of unresolved flows
  - Submits the event with the original business object, retrieves business object data before re-submitting, or discards the event from the system
  - Manages multiple InterChange Servers from one console
  - Queries by attribute value

- **Relationship Manager**
  - Monitors and manages the relationships between application objects and attributes to simultaneously synchronize data across multiple applications
  - Diagnoses and automatically cleans up data corruption situations when problems arise

- **System Manager**
  - Provides a visual interface for system administrators to monitor, control, and analyze the InterChange Server environment in a single user interface
  - Used to configure all InterChange Server components and can identify and correct system errors during runtime
  - Can start, stop, and pause all InterChange Server components as needed
  - Provides monitoring screens to track processing load by component
  - Provides traces and logs
  - Provides an audit trail for business events moving through the InterChange Server environment

- **System Monitor**
  Provides a centralized graphical and browser-based tool to monitor and control key system components such as adapters, maps, and collaborations within the WebSphere InterChange Server environment

- **Log Viewer**
  Displays trace information for business processes as they pass from one processing point to another
**WebSphere Business Integration Toolset for Development**

The WebSphere Business Integration Toolset for Development is used to configure, customize, or develop new and existing system components such as collaborations and maps. It consists of:

- **Process Designer**
  
  Process Designer is a visual business process modeling tool to graphically sketch and refine the logical flow of business processes. It uses UML-compliant graphical notation to represent the business process. It automatically generates pure Java code underneath the graphical notations and can extend the generated Java code to support complex business process modeling.

- **Test Environment**
  
  Test Environment supports testing both as stand-alone unit, Test Connector, or as an integrated testing environment.

- **Connector Configurator**
  
  The Connector Configurator is used for adding application-specific properties to a connector definition, for setting property values, and for configuring the connector definition with its business objects and maps.

- **Map Designer**
  
  Map Designer comes with the following features for specifying data transformations:
  
  - A visual drag and drop design tool to transform one type of business object to another type of business object
  - Build and customize transformation rules that convert data from application-specific formats into the business object model format of the InterChange Server
  - Uses the Activity Editor to generate transforms using a library of dates, strings, and math functions
  - Custom Java programming (optional)

- **Relationship Designer**
  
  The Relationship Designer defines the relationships between application objects and attributes to simultaneously synchronize data across multiple applications.

- **Business Object Designer**
  
  The Business Object Designer is a graphical tool for generating and maintaining business objects that allows analysts to rapidly create generic and application-specific business objects. It simplifies the creation of complex business objects using the Object Discovery Agents (ODA) feature by
inspecting a data source (JDBC, SAP, Siebel, XML, and so forth), and automatically generating the business object.

- **Object Discovery Agent Development Kit (ODK)**
  The ODK is an API that facilitates creation of Object Discovery Agents (ODAs). ODAs identify business object requirements specific to a data source and to generate definitions from those requirements.

- **Benchmarking Wizard**
  The Benchmarking Wizard enables you to test various IBM WebSphere components, interfaces, and systems to measure their throughput.

For information about each tool, visit the home page for the WebSphere Business Integration Toolset:

http://www.ibm.com/software/integration/wbitools/

### 1.5.10 WebSphere MQ Workflow

IBM WebSphere MQ Workflow is an IBM product aimed at helping organizations to automate their business processes. By introducing a process manager such as WebSphere MQ Workflow, the productivity of an organization can be improved. WebSphere MQ Workflow electronically enables an organization to deliver the right task, to the right person, at the right time, with the right data and the right tools. By automating the business processes, not only does WebSphere MQ Workflow help to optimize the organization’s resources, but it can reduce cycle time and associated costs. Using a process manager also decouples the process logic from the business logic and enables the possibility that a change in either the process logic or the business does not require a change in the other.

WebSphere MQ Workflow is best suited for automating process-centric business processes, as its strength lies in people-based workflows. WebSphere MQ Workflow supports many different staff delegation algorithms and it can also drive system integration via the implementation of one or more User Program Execution Servers (UPES). A UPES activity within a process sends a WebSphere MQ XML-formatted message to a user-defined WebSphere MQ queue. The UPES is a custom program that receives this message and performs the request, constructs an XML response that the WebSphere MQ Workflow server will understand, and sends the message back to the server. Since a UPES request is delivered by WebSphere MQ, this means that the UPES can run on any of the many operating system platforms that WebSphere MQ runs on.

WebSphere MQ Workflow can help to improve client satisfaction when the audit trail is used to provide status updates. The WebSphere MQ Workflow audit trail produces either an entry in the audit trail table in DB2 or an WebSphere MQ message in XML format for each workflow-relevant event that occurs. This audit
trail can provide enormous value to the business since process tracking can be made available, thereby allowing any interested parties in the process to determine where the process is. WebSphere MQ Workflow can be used with WebSphere Business Integration Workbench to provide real-time process tracking. Also, real production metrics from the audit trail can be used within WebSphere Business Integration Monitor to analyze an organization’s processes. Using these products together provides any client with the information necessary to achieve continuous process improvement.

WebSphere MQ Workflow is built upon proven IBM technology. The communication layer is built upon WebSphere MQ, while the database can use DB2. The servers can run on many OS platforms, including z/OS, providing flexibility for an organization’s environment. The operational model that can be developed for a WebSphere MQ Workflow implementation can be designed for highly available environments. The staff definitions kept within WebSphere MQ Workflow can be managed via the LDAP Bridge tool. Using the LDAP bridge eliminates the need for managing the staff manually.

WebSphere MQ Workflow contains the following components:

- **WebSphere MQ Workflow clients**
  
  With an MQ Workflow client or Web client, you can start and monitor the processes as they are defined in build time. If you are authorized, you can manage processes that are already running. The architecture of MQ Workflow allows you to use a standard MQ Workflow client, the MQ Web Workflow Web client, MQ Workflow Portal client, or a custom client created using APIs for the client functions.

- **WebSphere MQ Workflow server**
  
  The WebSphere MQ Workflow server provides a scalable and fault-tolerant runtime environment for business processes defined by WebSphere MQ Workflow Buildtime. It stores audit trails of running processes and sends notifications automatically to the persons specified in activities.

  The WebSphere MQ Workflow architecture allows you to manage your workload dynamically, depending on the setup you choose for your enterprise. Workload management is accomplished via pooling and MQ clustering.

  WebSphere MQ Workflow offers APIs to support the interaction between the WebSphere MQ Workflow server and client components. In addition, you can use APIs to invoke applications that you need for your workflow tasks.

- **WebSphere MQ Workflow Buildtime**
  
  Buildtime is the graphical process definition tool that is part of the WebSphere MQ Workflow product. You can graphically define business processes and
their activities to the level of detail needed for automation. Buildtime includes graphical support for declaring and documenting:

- Business rules on process navigation between steps
- Business rules for role-based work assignment
- “Need-to-know” constraints for process interactions with data flows and data mapping
- Process interface definitions (data, programs, queues)

► Web-services Process Management Toolkit

WebSphere MQ Workflow Web-services Process Management Toolkit is available as a SupportPac and provides the following capabilities:

- Compose Web services into a business process
  Composing Web services allows you to choreograph the interaction of a set of Web services within a business process and add control logic to the business process. Web services can be plugged together into a workflow process, allowing you to make the new e-business applications visible. With the toolkit you can choreograph Web services and even other software components, for example Java programs, to combine both intranet and Internet components.

- Implement a Web service as its own business process
  Using a process as the implementation for a Web service allows you to compose complex Web services with the characteristics of a process.

To implement a Web service as a business process you must create a process model and publish it as a Web service into a UDDI registry using the toolkit. As soon as the Web service is invoked, the process is executed under the control of the process management system of WebSphere MQ Workflow.

MQ Workflow offers a message-based XML interface that allows you to execute process instances. A process instance can invoke activity implementations by means of sending and receiving XML messages.

► Rapid Deployment Wizard

Available as a SupportPac, this tool allows you to quickly create JSP layout skeleton files for use with the WebSphere MQ Workflow Web client.

You can start the tool from IBM WebSphere Studio Application Developer. The tool gets its input from an FDL file (Flow Definition Language) that you can export from the WebSphere MQ Workflow Buildtime database. This tool enables you to create a JSP file for each program activity, including the putting and setting of fields corresponding to the data structure of each activity. You can edit the created JSP files with the WebSphere Studio Page...
Designer and publish them to a Web server on which the WebSphere MQ Workflow Web client is running. When you have created and published the JSP files, they are ready for use on the WebSphere MQ Workflow Web client.

- **ARIS to WebSphere MQ Workflow Bridge**

  The ARIS to WebSphere MQ Workflow Bridge (abbreviated as ARIS Bridge) is a SupportPac supplementary program to the IBM MQSeries Workflow software. It automatically converts the business process models designed with IDS Scheer ARIS Tool V5.0 into the FDL format supported by IBM MQSeries Workflow V3.2 or later. For instance, the bridge transfers process chains, organizational charts, and parts of the data model and the data flow. The conversion works in one direction: from ARIS Toolset to MQSeries Workflow. The standard installation of the ARIS Bridge works with a standard set of mapping rules that control the translation between ARIS and MQSeries Workflow models.

  The possible uses of this SupportPac are:
  - Conversion of ARIS business process models into MQSeries Workflow process models
  - Using the ARIS toolset as an improved development environment for Workflow modelling

  For further information, visit the home page for WebSphere MQ:
  
  http://www.ibm.com/software/integration/wmqwf

  **1.5.11 WebSphere Business Integration Connect**

  IBM WebSphere Business Integration Connect enables operational business to business (B2B) based on communities of trading partners. With this product, you can connect and integrate with communities of trading partners, improve interactions with suppliers and customers, and define trading partner interactions with an easy to use graphical interface. Typically, one enterprise will act as the hub and manage the trading community, while multiple community participants connect to this hub. The community participants can also act as hubs.

  IBM WebSphere Business Integration Connect provides:

  - Support for a wide range of industry-standard protocols including RosettaNet, AS2, EDI, and XML
  - Support for trading partner interactions over transports such as HTTP(S), FTP, SMTP and JMS/WebSphere MQ
  - Support for multiple security standards including third-party certificate authorities from VeriSign and Thawte, SSL support, and Non-repudiation as required for full AS2 compliance
1.5.12 WebSphere Data Interchange

WebSphere Data Interchange is a powerful data translation and transaction management solution that can grow with a client's electronic data interchange (EDI) and e-commerce requirements. The product translates EDI data, such as invoices, purchase orders, and billing forms for exchange with trading partners, and supports industry standards such as ANSI X12, EDIFACT, VICS, UCS, HIPAA, HL7, and RAIL. Translation can take place between any combination of EDI, XML, or structured application data format.

WebSphere Data Interchange allows businesses to leverage existing investments by distributing EDI messages with back-end systems, also eliminating manual processing steps and reducing cycle times.

WebSphere Data Interchange consists of:
- WebSphere Data Interchange Server
- WebSphere Data Interchange Client

For more information, visit the WebSphere Data Interchange home page:
http://www.ibm.com/software/integration/wdi/

1.5.13 WebSphere Business Integration Monitor

WebSphere Business Integration Monitor displays real-time data from events created by IBM WebSphere MQ Workflow to provide decision support for business performance management and optimization.

WebSphere Business Integration Monitor contains two primary components:
- Workflow Dashboard
- Business Dashboard

Workflow Dashboard
The Workflow Dashboard monitors the data and audit trail of WebSphere Business Integration Message Broker, WebSphere MQ Integrator Broker, and WebSphere MQ Workflow to provide an operational console of your company’s automated business processes. Specifically, process managers can track what business units or individuals might be under performing, which deadlines are in jeopardy, or other issues that potentially impede execution of corporate strategy.
A Web-based application, the Workflow Dashboard lets process managers perform any administrative or corrective action of in-flight work items from an Internet connection anywhere in the world. Upon making the corrective action, the Workflow Dashboard will then issue a command to WebSphere MQ Workflow via the appropriate APIs.

**Business Dashboard**
The Business Dashboard provides a higher-level, more strategic view of automated business processes than the Workflow Dashboard.

While the Workflow Dashboard provides intricate details on the automated business process, the Business Dashboard provides business statistics reporting by comparing actual company performance with targeted business goals. Additionally, the Business Dashboard locates and measures the cost of work items that match particular criteria. You can determine where your business stands against established milestones, and where shifts in business process execution could enhance company performance.

Actual metrics and statistical information from the Business Dashboard can be exported into WebSphere Business Integration Modeler for further analysis and enhancement. This data is vital to maintaining a realistic understanding of daily enterprise business performance. Without the ability to optimize the business process in a real-time environment, misuse of resources, delays, and process bottlenecks will affect the productivity of your organization.

For more information about Business Dashboard, visit the home page:


### 1.5.14 WebSphere Business Integration Modeler

WebSphere Business Integration Modeler is an Eclipse-based tool that helps business analysts model and simulate business processes graphically. Modeling a business process captures repeatable patterns of activities required to accomplish a business objective. Defining these patterns can help you improve business performance and to establish automation of the processes.

WebSphere Business Integration Modeler provides you with the ability to model business processes, business items such as documents and work products, resources such as employees or physical assets, and organizations, including structural relationships between entities in an organization. These processes can be analyzed and simulated.

When you work in the WebSphere Business Integration Modeler workbench you can select a user profile that reflects the level of detail required when working with models:
The Basic Business Modeling profile is intended for business analysts who want to work with a high-level view of a business process model.

The Intermediate Business Modeling profile is intended for providing more technical details of a model, including business rules and logic that apply to elements of the model.

The Advanced Business Modeling profile is intended for the development of detailed business process models that will be used as the basis for software applications.

The following technology modes are available for building business processes:

- **Business Process Execution Language (BPEL)**
  Business processes in BPEL format can be imported into WebSphere Studio Application Developer Integration Edition, where software application code can be built to implement the process.

- **Flow Definition Language (FDL)**
  Business processes built with FDL can be used in WebSphere MQ Workflow as the basis of an automated workflow solution.

- **Operational**
  This technology mode provides the ability to specify a greater level of detail than the previous two technology modes.

For more information, see the WebSphere Business Integration Modeler home page on the Web:


### 1.6 IBM WebSphere Commerce

IBM WebSphere Commerce provides all the functionality that you need to have a fully functional e-commerce site. It runs on industry leading products such as DB2 and WebSphere Application Server. It is designed for the security, scalability, and performance that any e-commerce site demands. It comes prepackaged with all the software required for a fully functional e-commerce site.

WebSphere Commerce V5.6 is available in the following editions:

- **Professional Edition**
  The Professional Edition provides tools and support for the following store model types:
  - Consumer direct store model (B2C)
Consumer direct supports commerce transactions involving products, services, or information between businesses and consumers. Consumers typically purchase goods or services directly from a business in a consumer direct scenario.

- B2B direct store model

  B2B direct supports commerce transactions involving products, services, or information between two businesses or parties. Typical B2B direct transactions occur between buyers, suppliers, manufacturers, resellers, distributors, and trading partners.

WebSphere Commerce Developer Professional Edition (formerly WebSphere Commerce Studio) provides a complete design and development environment for creating and maintaining e-commerce sites. This is purchased separately.

- Express

  WebSphere Commerce - Express is an affordable, yet complete solution aimed at mid-market companies. It features “all-in-one” licensing that gives you everything you need for production, staging, and site development. Like the Professional Edition, it can support both B2B and B2C store models. Unlike the Professional and Business editions, Express includes the WebSphere Commerce - Express Developer Edition development tool. Express comes with everything you need to get started quickly.

- Business Edition

  The Business Edition provides the capabilities of the Professional Edition, plus the following:

  - Advanced B2B direct

    In addition basic B2B direct store model support provided in the Professional Edition, B2B direct in the Business Edition also supports the extended site or Ex-Site, where a seller can have many sites aimed at different audiences. For example, a seller may create sites based on geographical regions, while some customized sites may be created for individual large clients. All these different sites can share assets such as the catalog, whereby each site selects the subset of the catalog that will be presented, and adjusts the prices as necessary.

  - Hosting

    The hosting model supports hosting of merchants or other businesses by an Internet Service Provider (ISP) or other hosting provider. Hosting models usually include a hub that provides self-provisioning tools that allow the merchant to create and administer a store, as well as tools that allow the hosting provider to manage all hosted stores.
Hosting providers may also include a site in which merchants can find and access the stores hosted by the provider. This is called a store directory.

- Demand chain

A demand chain is composed of the enterprises that sell a business’s goods or services. For example, a demand chain may be composed of buyers who initiate the sales transaction, the resellers who sell the manufacturer's goods, and the manufacturer who creates the goods. Or a demand chain may be composed of the resellers who sell a manufacturer's goods, the manufacturer who makes the goods, and the distributors who supply the manufacturer's goods to the resellers. Demand chains also support direct sales channels, in which the demand chain owner sells directly to consumers or business partners.

WebSphere Commerce Developer Business Edition (formerly WebSphere Commerce Studio) provides a complete design and development environment for creating and maintaining e-commerce sites. This is purchased separately.

For information about WebSphere Commerce, see:

- WebSphere Commerce Developer home page
- WebSphere Commerce Professional Edition home page
- WebSphere Commerce Business Edition home page
- WebSphere Commerce - Express home page
  http://www.ibm.com/software/genservers/commerce/express/

### 1.7 IBM WebSphere Portal for Multiplatforms

IBM WebSphere Portal provides a single point of access to applications, application content, processes, and people in your network. To provide this unified access, WebSphere Portal implements rich and valuable functions.

IBM WebSphere Portal allows you to establish customized portals for your employees, Business Partners, and clients. As illustrated in Figure 1-7 on page lxxi, the framework architecture implemented in this product provides a unified access point to internal and external Web applications, as well as portal access to other legacy applications. In this way, users sign on to the portal and receive personalized Web pages.
Figure 1-7   Horizontal and vertical portals

The personalized single point of access to all necessary resources reduces information overload, accelerates productivity, and increases Web site usage. In addition, portals do much more; for example they provide additional valuable functions such as security, search, collaboration, document management, document viewing, and workflow.

A portal delivers integrated content and applications, plus a unified, collaborative workplace. Indeed, portals are the next-generation desktop, delivering e-business applications over the Web to all kinds of client devices.

**Highlights and benefits**

IBM WebSphere Portal Version has been designed in response to the following set of fundamental business objectives:

- A single point of access to all resources associated with the portal domain.
- Personalized interaction with the portal services.
- Federated access to hundreds of data types and repositories, aggregated and categorized.
- Collaboration technologies that bring people together.
- Integration with applications, documents, and workflow system.

IBM and some industry analysts have coalesced around the concept of horizontal and vertical portals. Horizontal portals are the primary infrastructure upon which a portal is built. Vertical portals are built upon the horizontal layer and represent a specific portal instance, usually defined by a major topic or domain.
As illustrated in Figure 1-8, the horizontal portal infrastructure consists of several modular subsystems, including:

- **Presentation layer** - a Web user interface plus pervasive device support
- **Personalization** - the ability to serve dynamic response to the user based on personal profiles
- **Collaboration** - tools that allow e-mail, team rooms, shared places, and so forth to be exchanged
- **Portlets** - a framework for easily attaching software modules (portlets) and services
- **Applications and workflow** - integration of legacy and new applications
- **Search and categorize** - categorizing repositories of content and searching them for relevant information
- **Publish and subscribe** - the ability to author new content and publish it to subscribers
Chapter 1. IBM WebSphere product overview

- Administration and security - basic Web site services such as page designers, performance monitors, cluster services, and metadata management
- Integration - metadata sharing, XML, connectors, standards, EAI

WebSphere Portal provides additional services such as single sign-on, security, Web content publishing, search, personalization, Collaboration Services, enterprise application integration, support for mobile devices, and site analysis.

WebSphere Portal provides an extensible framework for interacting with enterprise applications, content, people, and processes. Self-service features allow end users to personalize and organize their own view of the portal, manage their own profiles, and publish and share documents with their colleagues.

WebSphere Portal comes in a variety of offerings tailored for the operating system environment and complexity of features required. For more information see the following:

- WebSphere Portal for Multiplatforms
  http://www.ibm.com/software/genservers/portal/
- WebSphere Portal Enable for iSeries
  http://www.ibm.com/software/genservers/portaliseries/
- WebSphere Portal Express for Multiplatforms
  http://www.ibm.com/software/genservers/portalexpress/
- WebSphere Portal for z/OS and OS/390
  http://www.ibm.com/software/genservers/portalzos/
WebSphere Application Server V5 architecture

WebSphere Application Server is IBM's implementation of the J2EE (Java 2 Enterprise Edition) platform, conforming to V1.3 of the specification. WebSphere Application Server is available in several different configurations that are designed to meet a wide range of customer requirements. Each configuration is packaged with other components that provide a unique environment. At the heart of each configuration is a WebSphere Application Server that provides the runtime environment for enterprise applications. This discussion will center around the runtime server component of the five flavors of WebSphere Application Server:

- IBM WebSphere Application Server - Express V5.1. We will refer to this as the Express configuration.
- IBM WebSphere Application Server V5.1. We will refer to this as the Base configuration.
- IBM WebSphere Application Server Network Deployment V5.1. We will refer to this as the Network Deployment configuration.
2.1 Server architecture

Each member of the WebSphere Application Server family uses essentially the same architectural structure. The simplest member, Express, contains a subset of the full architecture. The Base configuration is slightly more advanced; with Network Deployment at the top.

Note: The Enterprise configuration has been rebranded to WebSphere Business Integration Server Foundation and is discussed in WebSphere Business Integration Server Foundation Architecture and Overview, REDP9129.

2.1.1 Express configuration

The Express and Base configurations are similar in that they support a single-server environment. Although you can configure multiple application servers, there is no central administration or workload management. Each application server acts as an unique entity. The Express configuration is a slimmed down version of Base. As you will see, there are several differences between the two, but perhaps the most noticeable are the absence of the EJB container and embedded messaging support in Express.
2.1.2 Base configuration

The next step up in configurations is the Base configuration. Unlike the Express configuration, it has an EJB container, embedded messaging support, JCA resource adapter support and several other differences. The Base configuration has no central administration support and no workload management features.

Figure 2-2 shows an overview of the runtime architecture in a Base installation.
2.1.3 Network Deployment configuration

The Network Deployment configuration offers central administration and workload management. A Network Deployment environment consists of one or more Base installations and a Deployment Manager installation. The Base application servers are added to the cell and managed by the Deployment Manager. The Network Deployment package also includes the Web Services Private UDDI Registry and Web Services Gateway.
2.2 Cells, nodes and servers

Regardless of the configuration, the WebSphere Application Server is organized based on the concept of cells, nodes and servers. While all these elements are present in each configuration, cells and nodes do not play an important role until you reach the Network Deployment configuration.
Servers
Servers perform the actual code execution. There are several types of servers, depending on the configuration. Each server runs in its own JVM.

- **Application servers:**
  The application server is the primary runtime component in all configurations. This is where the application actually executes. All WebSphere Application Server configurations can have one or more application servers. In the Express and Base configurations, each application server functions as a separate entity. There is no workload distribution or common administration among application servers.

  In the Network Deployment configuration, multiple application servers are maintained from a central administration point. In addition, application servers can be clustered for workload distribution.

- **JMS servers:**
  The Base and Network Deployment configurations provide an embedded JMS server for messaging support. In the Base configuration, the JMS server functions are integrated into the application server. In the Network Deployment, the JMS server runs in a separate JVM. There is one JMS server per node.

Nodes (and node agents)
A node is a logical grouping of WebSphere-managed server processes that share common configuration and operational control. A node is generally associated with one physical installation of WebSphere Application Server. In the Express and Base configurations of WebSphere Application Server, there is only one node.

As you move up through the more advanced WebSphere Application Server configurations, the concepts of configuring multiple nodes from one common administration server and workload distribution among nodes are introduced. In these centralized management configurations, each node has a node agent that works with a Deployment Manager to manage administration processes.

Cells
A cell is a grouping of nodes into a single administrative domain. In the Base and Express configurations, a cell contains one node. That node may have multiple servers, but the configuration files for each server are stored and maintained individually.

With the Network Deployment configuration, a cell can consist of multiple nodes, all administered from a single point. The configuration and application files for all
nodes in the cell are centralized into a cell master configuration repository. This centralized repository is managed by the Deployment Manager process and synchronized out to local copies held on each of the nodes.

2.3 Servers

WebSphere Application Server uses servers to provide the functions required to host applications. Application servers execute user applications. They may act as individual servers or a member of a cluster. JMS servers provide the embedded messaging support.

Table 2-1  WebSphere Application Server server support

<table>
<thead>
<tr>
<th></th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application server</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JMS server</td>
<td>No</td>
<td>Yes (integrated into application server)</td>
<td>Yes</td>
</tr>
<tr>
<td>Application server clustering</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

2.3.1 Application server

Application servers provide the runtime environment for application code. They provide containers and services that specialize in enabling the execution of specific Java application components.

Each application server runs in its own Java virtual machine (JVM). In all configurations, one application server, called server1, is available automatically after installation. Additional application servers can be created, but unless you are using the Network Deployment configuration, the application servers are configured independently and have no workload distribution capabilities.

2.3.2 JMS server

Messaging support is provided through the use of one of the following JMS providers:

- WebSphere JMS provider (embedded)
- WebSphere MQ JMS provider
- Generic JMS provider

If you use the WebSphere MQ JMS provider or a generic JMS provider, the JMS functions are provided by an external product.

The embedded WebSphere JMS provider, on the other hand, is included with all WebSphere Application Server configurations except the Express configuration. The integrated messaging functions provided by the WebSphere JMS provider include support for message-driven beans, point-to-point and publish/subscribe styles of messaging, and integration with the transaction management service.

The functions of the embedded WebSphere JMS provider are implemented by a JMS server. In the Base configuration, the JMS server runs in the same JVM as the application server. From a configuration standpoint, it is a part of the application server.

In the Network Deployment configuration, the JMS server is separated from the application server and runs in a separate dedicated JVM. In this case, the configuration for the JMS server is independent of application servers.

### 2.3.3 Clusters

The Network Deployment configuration offers the option of using application server clustering to provide enhanced workload distribution. A cluster is a logical collection of application server processes, with the sole purpose of providing workload balancing.

Application servers that belong to a cluster are "members" of that cluster and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data.

For example, one cluster member might be running on a large multi-processor server while another member of that same cluster might be running on a small laptop. The server configuration settings for each of these two cluster members are very different, except in the area of application components assigned to them. In that area of configuration, they are identical.

The members of a cluster can be located on a single node (vertical cluster), across multiple nodes (horizontal cluster) or on a combination of the two.

**Application management**

When you install, update, or delete an application, the updates are automatically distributed to all members in the cluster.
Workload management

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS.

![Diagram of workload management](image)

This routing is based on weights associated with the cluster members. If all cluster members have identical weights, the plug-in will send equal requests to all members of the cluster assuming no strong affinity configurations. If the weights are scaled in the range from zero to twenty, the plug-in will route requests to those cluster members with the higher weight value more often. A rule of thumb formula for determining routing preference would be:

\[
\% \text{ routed to Server1} = \frac{\text{weight1}}{\text{weight1} + \text{weight2} + \ldots + \text{weightn}}
\]

Where there are \( n \) cluster members in the cluster.

The Web server plug-in will temporarily route around unavailable cluster members.

Workload management for EJB containers can be performed by configuring the Web container and EJB containers on separate application servers. Multiple application servers with the EJB containers can be clustered, enabling the distribution of EJB requests between the EJB containers.
In this configuration, EJB client requests are routed to available EJB containers in a round robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

The server weighted round robin routing policy will ensure a distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster would be that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism will send more requests to the higher weight value servers than the lower weight value servers. The policy ensures the desired distribution, based on the weights assigned to the cluster members. You can also choose to have requests sent to the node on which the client resides as the preferred routing. In this case, only cluster members on that node will be chosen (using the round robin weight method). Cluster members on remote nodes will only be chosen if a local server is not available.

2.4 Containers

The J2EE 1.3 specification defines the concept of containers to provide runtime support for applications. There are two containers in the application server implementation:

- Web container - Processes HTTP requests, servlets and JSPs
- EJB container - Processes enterprise beans (EJBs)

In addition, there is an application client container that can run on the client machine.
### Table 2-2  WebSphere Application Server container support

<table>
<thead>
<tr>
<th></th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web container</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EJB container</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Application client container</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### 2.4.1 Web container

The Web container processes servlets, JSP files and other types of server-side includes. Each application server runtime has one logical Web container, which can be modified, but not created or removed.

- **Servlet processing:** When handling servlets, the Web container creates a request object and a response object, then invokes the servlet service method. The Web container invokes the servlet's destroy method when appropriate and unloads the servlet, after which the JVM performs garbage collection.

- **Embedded HTTP server:** The Web container runs an embedded HTTP server for handling HTTP(S) requests from external Web server plug-ins or Web browsers. The embedded Web server is based on the IBM HTTP Server product.

  Directing client requests to the embedded Web server is useful for testing or development purposes and, in the Express configuration, can be considered for production use. In the more advanced configurations, the use of an external Web server and Web server plug-in as a front-end to the Web container is more appropriate for a production environment.

- **Session management:** Support is provided for the javax.servlet.http.HttpSession interface described in the Servlet API specification.

- **Web services engine:** Web services are provided as a set of APIs in cooperation with the J2EE applications. Web services engines are provided to support SOAP.

### Web server plug-ins

Although the Web container has an embedded HTTP server, a more likely scenario is that an external Web server will be used to receive client requests. The Web server can serve requests that do not require any dynamic content, for example, HTML pages. However, when a request requires dynamic content
(JSP/servlet processing), it must be forwarded to WebSphere Application Server for handling.

The mechanism to accomplish this is provided in the form of a Web server plug-in. The plug-in is included with the WebSphere Application Server packages for installation on a Web server. An XML configuration file, configured on the WebSphere Application Server, is copied to the Web server plug-in directory. The plug-in uses the configuration file to determine whether a request should be handled by the Web server or an application server. When a request for an application server is received, it is forwarded to the appropriate Web container in the application server. The plug-in can use HTTP or HTTPs to transmit the request.

2.4.2 EJB container

The EJB container provides all the runtime services needed to deploy and manage enterprise beans. It is a server process that handles requests for both session and entity beans.

The enterprise beans (packaged in EJB modules) installed in an application server do not communicate directly with the server; instead, the EJB container provides an interface between the EJBs and the server. Together, the container and the server provide the bean runtime environment.

The container provides many low-level services, including threading and transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained beans. A single container can host more than one EJB JAR file.

2.4.3 Client application container

The client application container is a separately installed component on the client's machine. It allows the client to run applications in an EJB-compatible J2EE environment.

There is a command-line executable (launchClient) that is used to launch the client application along with its client container runtime.
2.5 Application server services

The application server provides other services besides the containers.

Table 2-3  WebSphere Application Server services

<table>
<thead>
<tr>
<th>Service</th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCA services</td>
<td>Yes(^\d)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transaction service</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic cache service</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Message listener service</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ORB service</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Admin service (JMX)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Diagnostic trace service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Debugging service</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Name service (JNDI)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance Monitoring Interface (PMI) service</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security service (JAAS and Java 2 security)</td>
<td>JAAS but not Java 2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Business process engine</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

1. Included to provide JDBC support, but not configurable.
2.5.1 JCA services

Connection management for access to enterprise information systems (EIS) in WebSphere Application Server is based on the J2EE Connector Architecture (JCA) specification, also sometimes referred to as J2C. The connection between the enterprise application and the EIS is done through the use of EIS-provided resource adapters, which are plugged into the application server. The architecture specifies the connection management, transaction management, and security contracts that exist between the application server and EIS.

The Connection Manager in the application server pools and manages connections. It is capable of managing connections obtained through both resource adapters defined by the JCA specification and data sources defined by the JDBC 2.0 Extensions Specification.

Note that in the Express configuration, adding resource adapters is not supported. However, the JCA support is included to handle connectivity to JDBC resources. The connectivity is handled through a resource adapter provided by WebSphere Application Server.

2.5.2 Transaction service

WebSphere applications can use transactions to coordinate multiple updates to resources as one unit of work such that all or none of the updates are made permanent. Transactions are started and ended by applications or the container in which the applications are deployed.

WebSphere Application Server is a transaction manager that supports the coordination of resource managers through their XAResource interface and participates in distributed global transactions with other OTS 1.2 compliant transaction managers (for example J2EE 1.3 application servers).

WebSphere applications can also be configured to interact with databases, JMS queues, and JCA connectors through their local transaction support when distributed transaction coordination is not required.

The way that applications use transactions depends on the type of application component, as follows:

- A session bean can either use container-managed transactions (where the bean delegates management of transactions to the container) or bean-managed transactions (where the bean manages transactions itself).
- Entity beans use container-managed transactions.
- Web components (servlets) use bean-managed transactions.
In WebSphere Application Server, transactions are handled by three main components:

- A transaction manager that supports the enlistment of recoverable XAResources and ensures that each such resource is driven to a consistent outcome either at the end of a transaction or after a failure and restart of the application server.

- A container in which the J2EE application runs. The container manages the enlistment of XAResources on behalf of the application when the application performs updates to transactional resource managers (for example, databases). Optionally, the container can control the demarcation of transactions for enterprise beans configured for container-managed transactions.

- An application programming interface (UserTransaction) that is available to bean-managed enterprise beans and servlets. This allows such application components to control the demarcation of their own transactions.

### 2.5.3 Dynamic caching

The dynamic cache service improves performance by caching the output of servlets, commands and JSP files. The dynamic cache works within an application server, intercepting calls to cacheable objects, for example, through a servlet's service() method or a command's execute() method, and either stores the object's output to or serves the object's content from the dynamic cache.

Because J2EE applications have high read-write ratios and can tolerate small degrees of latency in the currency of their data, the dynamic cache can create an opportunity for significant gains in server response time, throughput, and scalability.

The following caching features are available in WebSphere Application Server.

- **Cache replication:**
  
  Cache replication among cluster members takes place using the WebSphere internal replication service. Data is generated one time and copied or replicated to other servers in the cluster, thus saving execution time and resources.

- **Cache disk offload:**
  
  By default, when the number of cache entries reaches the configured limit for a given WebSphere server, eviction of cache entries occurs, allowing new entries to enter the cache service. The dynamic cache includes an alternative feature named disk offload, which copies the evicted cache entries to disk for potential future access.
- **Edge Side Include caching:**
  The Web server plug-in contains a built-in ESI processor. The ESI processor has the ability to cache whole pages, as well as fragments, providing a higher cache hit ratio. The cache implemented by the ESI processor is an in-memory cache, not a disk cache, therefore, the cache entries are not saved when the Web server is restarted.

- **External caching:**
  The dynamic cache has the ability to control caches outside of the application server, such as IBM Edge Server, a non-z/OS IBM HTTP Server's FRCA cache, and a non-z/OS WebSphere HTTP Server plug-in ESI Fragment Processor. When external cache groups are defined, the dynamic cache matches externally cacheable cache entries with those groups, and pushes cache entries and invalidations out to those groups. This allows WebSphere to manage dynamic content beyond the application server. The content can then be served from the external cache, instead of the application server, improving savings in performance.

### 2.5.4 Message listener service

The message listener service is an extension to the JMS functions of the JMS provider. It provides a listener manager that controls and monitors one or more JMS listeners. Each listener monitors a JMS destination on behalf of a deployed message-driven bean.

### 2.5.5 Object Request Broker service

An Object Request Broker (ORB) manages the interaction between clients and servers, using IIOP. It enables clients to make requests and receive responses from servers in a network-distributed environment.

The ORB provides a framework for clients to locate objects in the network and call operations on those objects as if the remote objects were located in the same running process as the client, providing location transparency. The client calls an operation on a local object, known as a stub. Then the stub forwards the request to the desired remote object, where the operation is run and the results are returned to the client.

The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request on the network. The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client. The client-side ORB demarshals the returned results and passes the result to the
stub, which, in turn, returns to the client application, as if the operation had been run locally.

WebSphere Application Server uses an ORB to manage communication between client applications and server applications as well as communication among product components.

2.5.6 Name service

Each application server hosts a name service that provides a Java Naming and Directory Interface (JNDI) name space. The service is used to register resources hosted by the application server. The JNDI implementation in WebSphere Application Server V5 is built on top of a Common Object Request Broker Architecture (CORBA) naming service (CosNaming).

JNDI provides the client-side access to naming and presents the programming model used by application developers. CosNaming provides the server-side implementation and is where the name space is actually stored. JNDI essentially provides a client-side wrapper of the name space stored in CosNaming, and interacts with the CosNaming server on behalf of the client.

The naming architecture is used by clients of WebSphere applications to obtain references to objects related to those applications. These objects are bound into a mostly hierarchical structure, referred to as a name space. The name space structure consists of a set of name bindings, each consisting of a name relative to a specific context and the object bound with that name. The name space can be accessed and manipulated through a name server.

The following are features of a WebSphere Application Server V5 name space:

- The name space is distributed
  For additional scalability, the name space for a cell is distributed among the various servers. The Deployment Manager, node agent and application server processes all host a name server. In previous releases, there was only a single name server for an entire administrative domain.

  In WebSphere releases prior to V5.0, all servers shared the same default initial context, and everything was bound relative to that initial context. In WebSphere Application Server V5, the default initial context for a server is its server root. System artifacts, such as EJB homes and resources, are bound to the server root of the server with which they are associated.

- Transient and persistent partitions
  The name space is partitioned into transient areas and persistent areas. Server roots are transient. System-bound artifacts such as EJB homes and resources are bound under server roots. There is a cell persistent root, which
can be used for cell-scoped persistent bindings, and a node persistent root, which can be used to bind objects with a node scope.

► Federated name space structure

A name space is a collection of all names bound to a particular name server. A name space can contain naming context bindings to contexts located in other servers. If this is the case, the name space is said to be a federated name space, because it is a collection of name spaces from multiple servers. The name spaces link together to cooperatively form a single logical name space.

In a federated name space, the real location of each context is transparent to client applications. Clients have no knowledge that multiple name servers are handling resolution requests for a particular requested object.

In the Network Deployment configuration, the name space for the cell is federated among the Deployment Manager, node agents, and application servers of the cell. Each such server hosts a name server. All name servers provide the same logical view of the cell name space, with the various server roots and persistent partitions of the name space being interconnected by means of the single logical name space.

► Configured bindings

The configuration graphical interface and script interfaces can be used to configure bindings in various root contexts within the name space. These bindings are read-only and are bound by the system at server startup.

► Support for CORBA Interoperable Naming Service (INS) object URLs

WebSphere Application Server V5 contains support for CORBA object URLs (corbaloc and corbaname) as JNDI provider URLs and lookup names, as required by the J2EE 1.3 specification.

Figure 2-6 on page xciii summarizes the naming architecture and its components.
2.5.7 PMI service

WebSphere Application Server collects data on runtime and applications through the Performance Monitoring Infrastructure (PMI). This infrastructure is compatible with and extends the JSR-077 specification.

PMI uses a client-server architecture. The server collects performance data from various WebSphere Application Server components and stores it in memory. This data consists of counters such as servlet response time and data connection pool usage. The data can then be retrieved using a Web client, Java client or JMX client. WebSphere Application Server contains Tivoli Performance Viewer, a Java client which displays and monitors performance data.

WebSphere Application Server also collects data by timing requests as they travel through the product components. PMI request metrics log time spent in major components, such as Web containers, EJB containers, and databases. These data points are recorded in logs and can be written to Application Response Time (ARM) agents used by Tivoli monitoring tools.

Figure 2-6  Naming topology
The PMI services can be enabled for an application server or node agent using the administrative console. The type and level of data collected can be controlled through the administrative console (or wsadmin) or from the Tivoli Performance Viewer.

2.5.8 Security service

Each application server JVM hosts a security service that uses the security settings held in the configuration repository to provide authentication and authorization functionality.

2.5.9 Admin service

The admin service runs within each server JVM. In the Base configuration, the admin service runs in the application server. In the Network Deployment configuration, each of the following hosts an admin service:

- Deployment Manager
- Node agent
- Application server
- JMS server

The admin service provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository in the server's file system.

The admin service has a security control and filtering functionality, providing different levels of administration to certain users or groups using the following admin roles:

- Administrator
- Monitor
- Configurator
- Operator

2.6 Virtual hosts

A virtual host is a configuration enabling a single host machine to resemble multiple host machines. It allows a single physical machine to support several independently configured and administered applications. It is not associated with a particular node. It is a configuration, rather than a “live object”, which is why it can be created, but not started or stopped.
Each virtual host has a logical name and a list of one or more DNS aliases by which it is known. A DNS alias is the TCP/IP host name and port number used to request the servlet, for example yourHostName:80. When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet. If no match is found, an HTTP 404 error is returned to the browser.

IBM WebSphere Application Server provides two default virtual hosts:

- **default_host**
  
  Used for accessing most applications. For all configurations except Express, the default settings for default_host map to all requests for any alias on ports 80, 9443, and 9080. In the Express configuration, the default_host port settings are 80, 7443, and 7080. For example:
  
  http://localhost:80/servlet/snoop
  http://localhost:9080/servlet/snoop

- **admin_host**
  
  Specifically configured for accessing the WebSphere Application Server V5 administrative console. Other applications are not accessible through this virtual host. For all configurations except Express, the default settings for admin_host map to requests on ports 9090 and 9043. In the Express configuration, the default_host port settings are 7090, 7043. For example:
  
  http://localhost:9090/admin

### 2.7 Session management

In many Web applications, users dynamically collect data as they move through the site based on a series of selections on pages they visit. Where the user goes next, and what the application displays as the user's next page (or next choice) may depend on what the user has chosen previously from the site. In order to maintain this data, the application stores it in a “session”.

WebSphere supports three approaches to track sessions:

- **SSL session identifiers** - SSL session information is used to track the HTTP session ID.

- **Cookies** - The application server session support generates a unique session ID for each user, and returns this ID to the user's browser using a cookie. The default name for the session management cookie is JSESSIONID. Using cookies is the most common method of session management.

- **URL rewriting**
Session data can be kept in local memory cache, stored externally on a database, or kept in memory and replicated among application servers.

Table 2-4 shows the session support for each WebSphere Application Server configuration.

Table 2-4  WebSphere Application Server session management support

<table>
<thead>
<tr>
<th></th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookies</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>URL rewriting</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SSL session identifiers</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>In memory cache</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Session persistence using a database</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Memory-to-memory session persistence</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The Servlet 2.3 specification defines session scope at the Web application level, meaning that session information can only be accessed by a single Web application. However, there may be times when there is a logical reason for multiple Web applications to share information, for example a user name. WebSphere V5.0 provides an IBM extension to the specification allowing session information to be shared among Web applications within an enterprise application. This option is offered as an extension to the application deployment descriptor. No code change is necessary to enable this option. This option is specified during application assembling.

2.7.1 Session persistence

Many Web applications use the simplest form of session management, the in-memory local session cache. The local session cache keeps session information in memory and local to the machine and WebSphere Application Server where the session information was first created. Local session management does not share user session information with other clustered machines. Users only obtain their session information if they return to the
machine and WebSphere Application Server holds their session information on subsequent accesses to the Web site.

Most importantly, local session management lacks a persistent store for the sessions it manages. A server failure takes down not only the WebSphere instances running on the server, but also destroys any sessions managed by those instances.

By default, WebSphere places session objects in memory. However, with the Base and Network Deployment configurations, the administrator has the option of enabling persistent session management. This instructs WebSphere to place session objects in a persistent store. Using a persistent store allows the user session data to be recovered by another cluster member after a cluster member in a cluster fails or is shut down. Two options for session persistence are available:

- **Database**
  In a multi-server environment, session information can be stored in a central session database for session persistence. The multiple application servers hosting a particular application need to share this database information in order to maintain session states for the stateful components.

- **Memory-to-memory using WebSphere internal messaging**
  WebSphere internal messaging enables sessions to be shared among application servers without using a database. Using this method, sessions are stored in the memory of an application server, providing the same functionality as a database for session persistence. Two topology options are offered:
    - **Peer-to-peer topology**: Each application server stores sessions in its own memory. It also stores sessions to and retrieves sessions from other application servers. In other words, each application server acts as a client by retrieving sessions from other application servers, and each application server acts as a server by providing sessions to other application servers.
    - **Client/server topology**: Application servers act as either a replication client or a server. Those that act as replication servers store sessions in their own memory and provide session information to clients. They are dedicated replication servers that just store sessions but do not respond to the users’ requests. Client application servers send session information to the replication servers and retrieve sessions from the servers. They respond to user requests and store only the sessions of the users with whom they interact.

The internal messaging system is not associated with the embedded WebSphere JMS provider or JMS server. The necessary code is provided as part of the core WebSphere Application Server libraries.
2.8 Web services

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. WebSphere Application Server can act as both a Web service provider and as a requester. As a provider, it hosts Web services that are published for use by clients. As a requester, it hosts applications that invoke Web services from other locations.

WebSphere Application Server supports SOAP-based Web service hosting and invocation.

Table 2-5 WebSphere Application Server server support

<table>
<thead>
<tr>
<th></th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Web services support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Private UDDI Registry</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Web Services Gateway</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Enterprise Web services</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1 Not yet supported in V5.0

The Web services support includes the following:

- Web Services Description Language (WSDL), an XML-based description language that provides a way to catalog and describe services. It describes the interface of Web services (parameters and result), the binding (SOAP, EJB), and the implementation location.

- Universal Discovery Description and Integration (UDDI), a global, platform-independent, open framework to enable businesses to discover each other, define their interaction, and share information in a global registry.

- Simple Object Access Protocol (SOAP), a lightweight protocol for exchange of information in a decentralized, distributed environment.

- eXtensible Markup Language (XML), which provides a common language for exchanging information.

- Web Services Invocation Framework (WSIF), a runtime architecture for service-oriented applications that provides a binding-independent mechanism for Web service invocation. WSIF enables easy encapsulation of back-end
systems connected through the J2EE Connector Architecture (JCA) as Web services.

- JAX-RPC (JSR 101) provides the core programming model and bindings for developing and deploying Web services on the Java platform. It is a Java API for XML-based RPC and supports only JavaBeans as Web Service provider.

- Enterprise Web services (JSR 109): this adds EJBs and XML deployment descriptors to JSR 101.

- WS-Security: this specification covers a standard set of SOAP extensions that can be used when building secure Web services to provide integrity and confidentiality. It is designed to be open to other security models including PKI, Kerberos, and SSL. WS-Security provides support for multiple security tokens, multiple signature formats, multiple trust domains, and multiple encryption technologies. It includes security token propagation, message integrity, and message confidentiality. The specification is proposed by IBM, Microsoft, and VeriSign for review and evaluation. In the future, it will replace existing Web services security specifications from IBM and Microsoft including SOAP Security Extensions (SOAP-SEC), Microsoft's WS-Security and WS-License, as well as IBM's security token and encryption documents.

**Enterprise services (JCA Web services)**

Enterprise services offer access over the Internet to applications in a platform-neutral and language-neutral fashion. They offer access to enterprise information systems (EIS) and message queues and can be used in a client/server configuration without the Internet. Enterprise services can access applications and data on a variety of platforms and in a variety of formats.

An enterprise service wraps a software component in a common services interface. The software component is typically a Java class, EJB, or JCA resource adapter for an EIS. In services terminology, this software component is known as the implementation. Enterprise services primarily use WSDL and WSIF to expose an implementation as a service.

Using the Integrated Edition of WebSphere Studio, you can turn CICS and IMS transactions using Java Connector Architecture (JCA) into Web services.

**Web service client (requester)**

Applications that invoke Web services are known as Web service clients or Web service requestors. An application that acts as a Web service client is deployed to WebSphere Application Server like any other enterprise application. No additional configuration or software is needed for the Web services client to function. Web services clients can also be stand-alone applications.

A Web service client will bind to a Web service server to invoke the Web service. The binding is done using a service proxy (or stub), which is generated based on
the WSDL document for the Web service. This service proxy contains all the needed information to invoke the Web service and is used locally by the clients to access the business service. The binding can also be done dynamically using WSIF.

**Web service provider**
An application that acts as a Web service is deployed to WebSphere Application Server like any other enterprise application. The Web services are contained in Web modules or EJB modules.

Publishing the Web service to a UDDI registry makes it available to anyone searching for it. Web services can be published to a UDDI registry using the Web Services Explorer provided with WebSphere Studio Application Developer, Integration Edition, and Enterprise Developer configurations.

When using WebSphere Studio V5 to package the application for deployment, no additional configuration or software is needed for the Web services client to function. The SOAP servlets are automatically added and a SOAP admin tool is included in a Web module.

If not, you will need to use soapearenabler.bat, found in the WebSphere bin directory to enable the SOAP services within the EAR file and add the SOAP admin tool.

**Enterprise Web Services**
The Enterprise Web Services, based on the JSR 109 specification request, provides the use of JAX-RPC in a J2EE environment defining the runtime architecture, as well as the implementation and deployment of Web services in a generic J2EE server. The specification defines the programming model and architecture for implementing Web services in Java based on JSRs 67, 93, 101, and future JSRs related to Web services standards. The list of JSRs may be found at:

http://www.jcp.org/en/jsr/all

### 2.8.1 IBM WebSphere UDDI Registry
WebSphere Application Server V5 provides a private UDDI registry that implements V2.0 of the UDDI specification. This enables the enterprise to run its own Web services broker within the company or provide brokering services to the outside world.

The UDDI registry is installed as a Web application by the administrator on one application server (or cluster). A DB2 database is used to store registry data. In a test environment, Cloudscape can be used.
There are three interfaces to the registry for inquiry and publish:

- Through the UDDI SOAP API.
- Through the UDDI EJB client interface.
- Through the UDDI user console. This Web-based GUI interface can be used to publish and to inquire about UDDI entities but only provides a subset of the UDDI API functions.

Security for the UDDI registry is handled using WebSphere security. To support the use of secure access with the IBM WebSphere UDDI Registry, you need to configure WebSphere to use HTTPS and SSL.

### 2.8.2 Web Services Gateway

The Web Services Gateway bridges the gap between Internet and intranet environments during Web service invocations. The gateway builds upon the Web services Definition Language (WSDL) and the Web Services Invocation Framework (WSIF) for deployment and invocation.

The primary function of the Web Services Gateway is to map an existing WSDL-defined Web service to a new service that is offered by the gateway to others. The gateway thus acts as a proxy. External services are imported into the gateway and made available to the enterprise as internal proxy services. Likewise, internal services are imported into the gateway and made available as external proxy services. These services are also published to the relevant UDDI registries where required.

The Web Services Gateway is installed as Web application by the administrator.

**Exposing Web services to the outside world**

To expose an internal service for outside consumption, the gateway takes the WSDL file for the existing service and generates a new WSDL file that can be shared with outside requestors. The interface described in the WSDL is exactly the same, but the service endpoint is changed to the gateway, which is now the official endpoint for the service client.

You might also consider applying special security constraints to restrict the access to either specific users and/or specific methods of your service. This scenario is illustrated in Figure 2-7.
Importing Web services

An external service may be imported and made available as an internal service. This will help the internal service requestors invoke the service as if it were running on the gateway. Again, a new WSDL is generated by the gateway showing the same interface but naming the gateway as service provider rather than the real internal server. The gateway then reroutes all requests to the actual implementation specified in the original WSDL.

Of course, every client could access external Web services by traditional means, but if you add the gateway as an additional layer in between, clients do not have to change anything if the service implementor changes. This scenario is very similar to the illustration in Figure 2-7, with the difference that the client resides in the intranet and the Web service implementation is located at a site on the Internet.

Managing channels

A request for a service may originate in one protocol, but the service may be invoked in some other protocol by using the transformation function. Let us consider the case that an internal service is available on SOAP over JMS, but should be invoked using SOAP over HTTP. To handle such cases, you need to define a set of channels corresponding to the specific protocols on which you wish the Gateway to receive messages. The outbound channels over which the
Gateway invokes the target Web services are defined implicitly by the bindings in the WSDL imported in the Gateway and do not have to be explicitly configured. By using a different inbound channel to those defined in the imported WSDL bindings, you can switch the protocol.

**Managing filters**

You can develop and use one or more filters for your Web services, which means that some specific task is done using the input and/or output messages to and from your Web service. There are standard filters as well as the possibility of creating your own.

**UDDI publication and lookup**

The gateway facilitates working with UDDI registries. As you map a service for external consumption using the gateway, you can publish the exported WSDL in the UDDI registry. When the services in the gateway are modified, the UDDI registry is updated with the latest changes.

### 2.9 Security

IBM WebSphere Application Server security sits on top of the operating system security and the security features provided by other components, including the Java language.

- Operating system security protects sensitive WebSphere configuration files and authenticates users when the operating system user registry is used for authentication.

- Standard Java security is provided through the Java Virtual Machine (JVM) used by WebSphere and the Java security classes.

- The Java 2 Security API provides a means to enforce access control, based on the location of the code and who signed it. Java 2 security guards access to system resources such as file I/O, sockets, and properties. WebSphere global security settings allow you to enable or disable Java 2 security and provide a default set of policies. Java 2 security can be activated or inactivated independently from WebSphere global security; however, when global security is enabled, by default Java 2 security is also enabled.

The current principal of the thread of execution is not considered in the Java 2 security authorization. There are instances where it is useful for the authorization to be based on the principal, rather the code base and the signer. The Java Authentication and Authorization Services (JAAS) is a standard Java API that allows the Java 2 security authorization to be extended to the code base on the principal as well as the code base and signers.
JAAS is a standard extension to the Java 2 SDK v1.3 and is part of the Java 2 SDK V1.4. The JAAS programming model allows the developer to design application authentication in a pluggable fashion, which makes the application independent from the underlying authentication technology. JAAS does not require Java 2 security to be enabled.

► The Common Secure Interoperability protocol adds additional security features that enable interoperable authentication, delegation and privileges in a CORBA environment. It supports interoperability with the EJB 2.0 specification and can be used with SSL.

► J2EE security uses the security collaborator to enforce J2EE-based security policies and support J2EE security APIs. APIs are accessed from WebSphere applications in order to access security mechanisms and implement security policies. J2EE security guards access to Web resources such as servlets/JSPs and EJB methods based on roles defined by the application developer. Users and groups are assigned to these roles during application deployment.

► IBM Java Secure Socket Extension (JSEE) is the Secure Sockets Layer (SSL) implementation used by WebSphere Application Server. It is a set of Java packages that enable secure Internet communications. It implements a Java version of SSL and Transport Layer Security (TLS) protocols and includes functionality for data encryption, server authentication, message integrity, and client authentication.

WebSphere Application Server V5 security relies on and enhances all the above mentioned layers. It implements security policy in a unified manner for both Web and EJB resources.

Table 2-6 on page civ shows the security features supported by the WebSphere Application Server configurations.

| Table 2-6  WebSphere Application Server security support |
|------------|--------|--------|--------|
|            | Express | Base   | ND     |
| Java 2 security | No      | Yes    | Yes    |
| J2EE security (role mapping) | Yes | Yes    | Yes    |
| JAAS        | Yes    | Yes    | Yes    |
| CSIv2       | No     | Yes    | Yes    |
| Security authentication | LDAP, SWAM | LDAP, SWAM | LDAP  |
Figure 2-8 presents a general view of the logical layered security architecture model of WebSphere Application Server. The flexibility of that architecture model lies in pluggable modules that can be configured according to the requirements and existing IT resources.

<table>
<thead>
<tr>
<th>User registry</th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local OS, LDAP, Custom Registry</td>
<td>Local OS, LDAP, Custom Registry</td>
<td>Local OS, LDAP, Custom Registry</td>
<td></td>
</tr>
</tbody>
</table>

2.9.1 User registry

The pluggable user registry allows you to configure different databases to store user IDs and passwords that are used for authentication and authorization. There are three options:

- Local operating system user registry
  When configured, WebSphere uses the operating system’s users and groups for authentication.

- LDAP user registry
  In many solutions, LDAP user registry is recommended as the best solution for large scale Web implementations. Most of the LDAP servers available on the market are well equipped with security mechanisms that can be used to
securely communicate with WebSphere Application Server. The flexibility of search parameters that an administrator can set up to adapt WebSphere to different LDAP schemas is considerable.

- Custom user registry
  This leaves an open door for any custom implementation of a user registry database. WebSphere API provides the UserRegistry Java interface that you should use to write the custom registry. This interface may be used to access virtually any relational database, flat files and so on.

Only one single registry can be active at a time.

2.9.2 Authentication

Authentication is the process of establishing whether a client is valid in a particular context. A client can be either an end user, a machine, or an application. The pluggable authentication module allows you to choose whether WebSphere will authenticate the user or will accept the credentials from external authentication mechanisms.

An authentication mechanism in WebSphere typically collaborates closely with a user registry when performing authentication. The authentication mechanism is responsible for creating a credential which is a WebSphere internal representation of a successfully authenticated client user. Not all credentials are created equal. The abilities of the credential are determined by the configured authentication mechanism.

Although WebSphere provides several authentication mechanisms, only a single “active” authentication mechanism can be configured at once. The active authentication mechanism is selected when configuring WebSphere global security.

WebSphere provides two authentication mechanisms: Simple WebSphere Authentication Mechanism (SWAM) and Lightweight Third Party Authentication (LTPA). These two authentication mechanisms differ primarily in the distributed security features each supports.

- Simple WebSphere Authentication Mechanism (SWAM)
  The SWAM authentication mechanism is intended for simple, non-distributed, single application server type runtime environments. The single application server restriction is due to the fact that SWAM does not support forwardable credentials. What this means is that if a servlet or EJB in application server process 1 invokes a remote method on an EJB living in another application server process 2, the identity of the caller identity in process 1 is not transmitted to server process 2. What is transmitted is an unauthenticated
credential, which, depending on the security permissions configured on the EJB methods, may cause authorization failures.

Since SWAM is intended for a single application server process, single sign-on (SSO) is not supported.

The SWAM authentication mechanism is suitable for simple environments, software development environments, or other environments that do not require a distributed security solution.

SWAM relies on the session ID; it is not as secure as LTPA, therefore using SSL with SWAM is strongly recommended.

- **Light Weight Third Party Authentication (LTPA)**

Lightweight Third Party Authentication (LTPA) is intended for distributed, multiple application servers and machine environments. It supports forwardable credentials and SSO. LTPA is able to support security in a distributed environment through the use of cryptography. This allows LTPA to encrypt, digitally sign and securely transmit authentication related data and later decrypt and verify the signature.

LTPA requires that the configured user registry be a central shared repository such as LDAP or a Windows domain type registry.

### 2.9.3 Authorization

Pluggable authorization interfaces will allow the use of different authorization mechanisms for WebSphere applications. In the current version, JAAS is supported and Tivoli Access Manager is an external authorization system.

WebSphere Application Server standard authorization mechanisms are based on the J2EE security specification and Java Authentication and Authorization Services (JAAS).

- **Java 2 security architecture** uses a security policy to specify who is allowed to execute code in the application. Code characteristics, like a code signature, signer ID, or source server, determine whether or not the code will be granted access to be executed.

- **JAAS** extends this approach with role-based access control. Permission to execute a code is granted not only based on the code characteristics but also on the user, who is running it. JAAS programming models allow the developer to design application authentication in a pluggable fashion, which makes the application independent from the underlying authentication technology.

For each authenticated user, a Subject class is created and a set of Principals is included in the subject in order to identify that user. Security policies are granted based on possessed principals.
2.9.4 Security components

Figure 2-9 shows an overview of the security components that come into play in WebSphere Application Security.

![Diagram showing security components in WebSphere Application Security]

**Security server**

The security server is a component of WebSphere Application Server that runs in each application server process. If multiple application server instances are executed on a single node, then multiple security servers exist on that node.

The security server component is responsible for managing authentication and for collaborating with the authorization engine and the user registry.

**Security collaborators**

Security collaborators are application server processes responsible for enforcing security constraints specified in deployment descriptors. They communicate with the security server every time that authentication and authorization actions are required. The following security collaborators are identified.

- **Web security collaborator**

  The Web security collaborator resides in the Web container and provides the following services to the application:
  - Checks authentication
  - Performs authorization according to the constraint specified in the deployment descriptor
– Logs security tracing information

➤ EJB security collaborator

The EJB security collaborator resides in the EJB container. It uses CSIv2 and SAS to authenticate Java client requests to enterprise beans. It works with the security server to perform the following functions:
– Check authorizations according to the specified security constraint
– Support communication with local user registry
– Log security tracing information
– Communicate external ORB using CSIv2 when a request for a remote bean is issued

2.9.5 Security flows

The following sections outline the general security flow.

Web browser communication

The following steps describe the interaction of the components from a security point of view when a Web browser sends a request to a WebSphere application.

1. The Web user requests a Web resource protected by WebSphere Application Server.

2. The Web server receives the request and recognizes that the requested resource is on the application server, and, using the Web server plug-in, redirects the request.

3. The Web server plug-in passes the user credentials to the Web security collaborator, which performs user authentication.

4. After successful authentication, the Web request reaches the Web container. The Web security collaborator passes the user’s credentials and the security information contained in the deployment descriptor to the security server for authorization.

5. Upon subsequent requests, authorization checks are performed either by the Web collaborator or the EJB collaborator, depending on what the user is requesting. User credentials are extracted from the established security context.
**Administrative tasks**

Administrative tasks are issued using either the Web-based administrative console or the wsadmin scripting tool. The following steps illustrate how the administration tasks are executed.

1. The administration client generates a request that reaches the server side ORB and JMX MBeans. The JMX MBeans represent managed resources.

2. The JMX MBeans contact the security server for authentication purposes. JMX beans have dedicated roles assigned and do not use user registry for authentication and authorization.

**Java client communication**

The steps below describe how a Java client interacts with a WebSphere application.

1. A Java client generates a request that reaches the server side ORB.

2. The CSIv2 or IBM SAS interceptor performs authentication on the server side on behalf of the ORB, and sets the security context.

3. The server side ORB passes the request to the EJB container.

4. After submitting a request to the access protected EJB method, the EJB container passes the request to the EJB collaborator.

5. The EJB collaborator reads the deployment descriptor from the .ear file and user credential from the security context.

6. Credentials and security information is passed to the security server which validates user access rights and passes this information back to the collaborator.

7. After receiving a response from the security server, the EJB collaborator authorizes or denies access to the user to the requested resource.

### 2.10 Resource providers

Resource providers define resources needed by running J2EE applications. Table 2-7 shows the resource provider support of the WebSphere Application Server configuration.

<table>
<thead>
<tr>
<th></th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDBC provider</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JavaMail</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### 2.10.1 JDBC resources

A data source represents a real-world data source, such as a relational database. When a data source object has been registered with a JNDI naming service, an application can retrieve it from the naming service and use it to make a connection to the data source it represents.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of properties on the DataSource object. This makes an application more portable because it does not need to hard code a driver name, which often includes the name of a particular vendor. It also makes maintaining the code easier because if, for example, the data source is moved to a different server, all that needs to be done is to update the relevant property in the data source. None of the code using that data source needs to be touched.

Once a data source has been registered with an application server’s JNDI name space, application programmers can use it to make a connection to the data source it represents.

The connection will usually be a pooled connection, that is, once the application closes the connection, the connection is returned to a connection pool, rather than being destroyed.

Data source classes and JDBC drivers are implemented by the data source vendor. By configuring a JDBC provider, we are providing information about the set of classes used to implement the data source and the database driver, that is, it provides the environment settings for the DataSource object.

WebSphere Application Server V5 provides two types of data sources, each differentiated by how the connections are handled:

- WebSphere Version 4 data source
- WebSphere Version 5 data source
**Version 4 data source**

WebSphere Version 4.0 provides its own JDBC connection manager to handle connection pooling and JDBC access. This support is included with WebSphere Application Server V5 to provide support for J2EE 1.2 applications. If an application chooses to use a Version 4 data source, the application will have the same connection behavior as in WebSphere Version 4.

![Connection pooling in WebSphere Version 4](image)

**Figure 2-10  Connection pooling in WebSphere Version 4**

**WebSphere 5 data source**

In WebSphere Application Server V5, connection pooling is provided by two parts, a JCA Connection Manager and a relational resource adapter.
The JCA Connection Manager provides the connection pooling, local transaction, and security supports. The relational resource adapter provides both JDBC wrappers and JCA CCI implementation that allows BMP, JDBC applications and CMP beans to access the database.

2.10.2 JavaMail

The JavaMail APIs provide a platform and protocol-independent framework for building Java-based mail client applications. The JavaMail APIs require service providers, known in WebSphere as protocol providers, to interact with mail servers that run the pertaining protocols.

A JavaMail provider encapsulates a collection of protocol providers. WebSphere Application Server has a Built-in Mail Provider that encompasses three protocol providers: SMTP, IMAP and POP3. These protocol providers are installed as the default and should be sufficient for most applications.

- Simple Mail Transfer Protocol (SMTP). This is a popular transport protocol for sending mail. JavaMail applications can connect to an SMTP server and send mail through it by using this SMTP protocol provider.
- Post Office Protocol (POP3). This is the standard protocol for receiving mail.
- Internet Message Access Protocol (IMAP). This is an alternative protocol to POP3 for receiving mail.
To use other protocols, you must install the appropriate service provider for those protocols.

In addition to service providers, JavaMail requires the JavaBeans Activation Framework (JAF) as the underlying framework to deal with complex data types that are not plain text, like Multipurpose Internet Mail Extensions (MIME), Uniform Resource Locator (URL) pages, and file attachments.

The JavaMail APIs, the JAF, the service providers and the protocols are shipped as part of WebSphere Application Server using the following Sun licensed packages:

- **mail.jar**: Contains the JavaMail APIs, and the SMTP, IMAP, and POP3 service providers.
- **activation.jar**: Contains the JavaBeans Activation Framework.
- **Application Server - Express**: Supports JavaMail V1.2 and the JAF V1.0.

### 2.10.3 JCA resource adapters

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting the J2EE platform to heterogeneous Enterprise Information Systems (EIS), for example ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language.

The JCA resource adapter is a system-level software driver supplied by EIS vendors or other third-party vendors. It provides the connectivity between J2EE components (an application server or an application client) and an EIS.

To use a resource adapter, you need to install the resource adapter code and create connection factories that use the adapter.

One resource adapter, the WebSphere Relational Resource Adapter, is predefined for handling data access to relational databases. This resource adapter provides data access through JDBC calls to access databases dynamically. It provides connection pooling, local transaction, and security support. The WebSphere persistence manager uses this adapter to access data for container-managed persistence (CMP) beans.

### 2.10.4 URL providers

URL providers implement the functionality for a particular URL protocol, such as HTTP, by extending the java.net.URLStreamHandler and java.net.URLConnection classes. It enables communication between the application and a URL resource that is served by that particular protocol.
A URL provider named Default URL Provider is included in the initial WebSphere configuration. This provider utilizes the URL support provided by the IBM JDK. Any URL resource with protocols based on the Java 2 Standard Edition 1.3.1, such as HTTP, FTP or File, can use the default URL provider.

Customers can also “plug in” their own URL providers that implement other protocols not supported by the JDK.

### 2.10.5 JMS providers

The JMS functionality provided by WebSphere includes support for three types of JMS provider:

- WebSphere JMS provider (embedded messaging)
- WebSphere MQ JMS provider
- Generic JMS providers

There can be more than one JMS provider per node. That is, a node can be configured to concurrently make use of any combination (or all) of the WebSphere JMS provider, WebSphere MQ JMS provider and a generic JMS provider.

There can only be one JMS server process (embedded WebSphere JMS provider) per node. However, it is possible to have both the WebSphere JMS provider and full WebSphere MQ concurrently installed on the same machine. In this case, there will only be one copy of the JMS client classes (MA88) and MQ transport code (binaries). Because of this, both must be at the same version and patch level.

The support provided by WebSphere administration tools for configuration of JMS providers differs depending upon the provider. Table 2-8 provides a summary of the support.

**Table 2-8  WebSphere administration support for JMS provider configuration**

<table>
<thead>
<tr>
<th>Configurable objects</th>
<th>WebSphere JMS provider</th>
<th>WebSphere MQ JMS provider</th>
<th>Generic JMS provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial context factory and provider URL</td>
<td>N **</td>
<td>N **</td>
<td>Y</td>
</tr>
<tr>
<td>Messaging system objects (queues/topics)</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>JMS administered objects (JMS connection factory and JMS destination)</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

** The settings are not exposed in the WebSphere administrative console.
WebSphere JMS provider (embedded messaging)

The embedded messaging support is provided as a WebSphere Application Server installation option. The embedded support provides both point-to-point and publish/subscribe functions.

This support, referred to as the WebSphere JMS provider, is composed of the following IBM products:

- WebSphere MQ
- WebSphere MQ MA88 SupportPac - JMS client classes
- An early release of WebSphere Event Broker, providing the publish/subscribe service

Each of these is reduced in footprint and function compared to the independently available product.

The WebSphere JMS provider supports the requirements of the J2EE 1.3 specification:

- Support for the JMS 1.0.2 specification
- Providing an implementation of JMS Server 1.0
- Full compliance with the J2EE 1.3 compliance tests

The WebSphere JMS provider does not support:

- APIs other than those defined in the JMS API
- Interoperability with WebSphere MQ queue managers

Security is integrated with the WebSphere security system, with user and group access being configured using the WebSphere administration tools.

The JMS server hosts the broker and manages the external WebSphere MQ processes, including creation, management and deletion of the WebSphere MQ queues and topics.

All JMS client access is performed using the WebSphere MQ client (TCP/IP) mode.

WebSphere MQ JMS provider

WebSphere Application Server V5 supports the use of full WebSphere MQ as the JMS provider. The product is tightly integrated with the WebSphere installation, with WebSphere providing the JMS client classes and administration interface, while WebSphere MQ provides the queue-based messaging system.

Generic JMS providers

WebSphere Application Server V5 supports the use of generic JMS providers, as long as they implement the ASF component of the JMS 1.0.2 specification. JMS
resources for generic JMS providers are not configurable using WebSphere administration.

2.10.6 Resource environment providers

The java:comp/env environment provides a single mechanism by which both JNDI name space objects and local application environment objects can be looked up. WebSphere Application Server provides a number of local environment entries by default.

The J2EE 1.3 specification also provides a mechanism for defining custom (non-default) environment entries using <resource-env-ref> entries defined in an application's standard deployment descriptors. The J2EE 1.3 specification separates the definition of the resource environment entry from the application by:

1. Requiring the application server to provide a mechanism for defining separate administrative objects that encapsulate a resource environment entry. The administrative objects are to be accessible via JNDI in the application server's local name space (java:comp/env).

2. Specifying the administrative object's JNDI lookup name and the expected returned object type in <resource-env-ref>.

WebSphere Application Server supports the <resource-env-ref> mechanism by providing administration objects for the following:

- Resource environment provider
  Defines an administrative object that groups together the referenceable, resource environment entry administrative objects and any required custom properties.

- Referenceable
  Defines the class name of the factory class that returns object instances implementing a Java interface.

- Resource environment entry
  Defines the binding target (JNDI name), factory class and return object type (via the link to the referenceable) of the resource environment entry.

2.11 Administration

IBM WebSphere Application Server V5 provides a new administration model based on the JMX framework. JMX allows you to wrap hardware and software resources in Java and expose them in a distributed environment. JMX also
provides a mapping framework for integrating existing management protocols, such as SNMP, into JMX's own management structures.

Each application server has an administration service that provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository. The repository is actually a set of XML files stored in the server's file system.

2.11.1 Administration tools

Table 2-9 shows the administration tool support for WebSphere Application Server by configuration.

<table>
<thead>
<tr>
<th></th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative console</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Commands</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>wsadmin</td>
<td>Included, but not recommended</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Administrative console**

The administrative console is a Web-browser based interface that provides configuration and operation capability. It is implemented with an enterprise application (adminconsole.ear) installed on an application server. The administrator connects to the application using a Web browser client.

Users assigned to different administration roles can manage the application server and certain components and services using this interface. Because the administrative console is an application, the server and application must both be started before you can use it.

In the Express and Base configurations, the adminconsole application is installed on server1. If you create additional servers, you can configure them using the administrative console application on server1 but will have no runtime capabilities. You can install the adminconsole.ear application on each server in order to have full operational capabilities for that server.

In the Network Deployment configuration, the administrative console application is installed and runs on the Deployment Manager. When a node is added to a cell, the adminconsole application is deleted from the node and the configuration
files are integrated into the master cell repository to be maintained by the Deployment Manager.

**Commands**
WebSphere Application Server provides a set of commands in the `<server_install>/bin` directory that allow you to perform a subset of administrative functions. For example, the `startServer` command is provided to start an application server.

**wsadmin scripting client**
The `wsadmin` scripting client provides extra flexibility over the Web-based administration application, allowing administration using the command-line interface. Using the scripting client not only makes administration quicker, but helps automate the administration of multiple application servers and nodes using scripts.

The scripting client uses the Bean Scripting Framework (BSF), which allows a variety of scripting languages to be used for configuration and control.

The `wsadmin` scripting interface is included in all WebSphere Application Server configurations but is targeted toward advanced users. The use of `wsadmin` requires in-depth familiarity with application server architecture and a scripting language. It is not recommended for WebSphere Application Server - Express users.

### 2.11.2 Configuration repository
The configuration repository holds copies of the individual component configuration documents. Unlike previous versions of WebSphere Application Server, which used a relational database to hold configuration information, in V5 all configuration information is stored in XML files. The application server's Admin service takes care of the configuration and makes sure it is consistent during the runtime.

### 2.11.3 Centralized administration
The Network Deployment configuration allows multiple servers and nodes to be administered from a central location. This is facilitated through the use of a central Deployment Manager that handles the administration process and distributes the updated configuration to the node agent for each node. The node agent, in turn, is responsible for maintaining the configuration for the servers in the node.
Table 2-10  WebSphere Application Server distributed administration support

<table>
<thead>
<tr>
<th></th>
<th>Express</th>
<th>Base</th>
<th>ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment Manager</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Node agent</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Managed processes**
All operating system processes that are components of the WebSphere product are called managed servers or managed processes. JMX support is embedded in all managed processes and these processes are available to receive administration commands and to output administration information about the state of the managed resources within the processes.

WebSphere provides the following managed servers/processes:

- **Deployment Manager**
  Provides a single point to access all configuration information and control for a cell. The Deployment Manager aggregates and communicates with all of the node agent processes on each node in the system.

- **Node agent**
  Aggregates and controls all of the WebSphere managed processes on its node. There is one node agent per node.

- **Application server**
  Managed server that hosts J2EE applications.

- **JMS server**
  Managed server that hosts the embedded messaging service for a node. There is one JMS server per node. In a base configuration, the JMS server functions are provided within the application server process.

**Deployment Manager**
The Deployment Manager process provides a single, central point of administrative control for all elements in the cell. It hosts the Web-based administrative console application. Administrative tools that need to access any managed resource in a cell usually connect to the Deployment Manager as the central point of control.

The Deployment Manager is responsible for the content of the repositories (configuration and application binaries) on each of the nodes. It manages this through communication with the node agent process resident on each node of the cell.
Using the Deployment Manager, horizontal scaling, vertical scaling and distributed applications are all easy to administer and manage, since application servers are managed by nodes, and one or more nodes is/are managed by a cell.

**Node agent**
The node agent is an administrative process and is not involved in application serving functions. It hosts important administrative functions such as:

- File transfer services
- Configuration synchronization
- Performance monitoring

The node agent aggregates and controls all the managed processes on its node by communicating with:

- The Deployment Manager to coordinate configuration synchronization and to perform management operations on behalf of the Deployment Manager.
- Application servers and JMS servers to manage (start/stop) each server and to update its configuration and application binaries as required.

Only one node agent is defined (and run) on each node. In an Express or Base installation, there is no node agent.

**Master configuration repository**
With the Network Deployment configuration, a master configuration repository that contains all of the cell's configuration data is maintained by the Deployment Manager. The configuration repository at each node is a synchronized subset of the master repository. The node repositories are read-only for application server access, since only the Deployment Manager can initiate their update, pushing configuration changes out from the cell master configuration repository.

### 2.12 The flow of an application

Figure 2-12 on page cxxii shows the typical application flow for Web browser clients using either JDBC (from a servlet) or EJB to access application databases.
Figure 2-12 Application flow

Figure 2-12 is explained below:

1. A Web client requests a URL in the browser (input page).
2. The request is routed to the Web server over the Internet.
3. The Web server immediately passes the request to the Web server plug-in. All requests go to the WebSphere plug-in first.
4. The Web server plug-in examines the URL, verifies the list of hostname aliases from which it will accept traffic based on the virtual host information, and chooses a server to handle the request.
5. A stream is created. A stream is a connection to the Web container. It is possible to maintain a connection (stream) over a number of requests. The Web container receives the request and, based on the URL, dispatches it to the proper servlet.
6. If the servlet class is not loaded, the dynamic class loader loads the servlet (servlet init(), then doGet() or doPost()).
7. JNDI is now used for lookup of either datasources or EJBs required by the servlet.
8. Depending upon whether a datasource is specified or an EJB is requested, the JNDI will direct the servlet:
   a. To the corresponding database, and get a connection from its connection pool in the case of a data source.
b. To the corresponding EJB container, which then instantiates the EJB when an EJB is requested

9. If the EJB requested involves an SQL transaction, it will go back to the JNDI to look up the datasource.

10. The SQL statement will be executed and the data retrieved will be sent back:
   a. To the servlet
   b. To the EJB

11. Data beans are created and handed off to JSPs in the case of EJBs.

12. The servlet sends data to JSPs.

13. The JSP generates the HTML that is sent back through the WebSphere plug-in to the Web server.

14. The Web server sends the output page (output HTML) to the browser.

2.13 Developing and deploying applications

![Diagram of application development process]

**Application design**
Design tools like Rational Rose or Rational XDE can be used to model the application using the Unified Modeling Language (UML). The output of the
modeling will generally consist of use case scenarios, class diagrams, and starter code generated based on the model.

**Application development**

Application development is done using WebSphere Studio (or a comparable IDE) to create the enterprise application. Use the WebSphere Studio configuration that most closely matches your needs and the capabilities of the WebSphere Application Server configuration to build the enterprise application.

You can start by importing pre-generated code such as from Rational Rose, a sample application, an existing production application, or you can start from scratch.

WebSphere Studio provides many tools and aids to get you started quickly. WebSphere Studio supports team development using CVS or Rational ClearCase, allowing multiple developers to share a single master source copy of the code.

During the development phase, component testing can be done using the built-in WebSphere Application Server test environment. WebSphere Studio provides server tools capable of creating and managing servers both in the test environment and on remote server installations. The application is automatically packaged into an EAR file for deployment when you run the application on a server using WebSphere Studio.

**Application packaging**

J2EE 1.3 applications are packaged into Enterprise Application Archive (EAR) files to be deployed to one or more application servers. A J2EE 1.3 application contains any or all of the modules shown in Table 2-11.

<table>
<thead>
<tr>
<th>Module</th>
<th>Filename</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web module</td>
<td>&lt;module&gt;.war</td>
<td>Servlets, JSP files, and related code artifacts</td>
</tr>
<tr>
<td>EJB module</td>
<td>&lt;module&gt;.jar</td>
<td>Enterprise beans and related code artifacts</td>
</tr>
<tr>
<td>Application client module</td>
<td>&lt;module&gt;.jar</td>
<td>Application client code</td>
</tr>
<tr>
<td>Resource adapter module</td>
<td>&lt;module&gt;.rar</td>
<td>Library implementation code that your application uses to connect to enterprise information systems (EIS)</td>
</tr>
</tbody>
</table>

This packaging is done automatically in WebSphere Studio when you export an application for deployment. If you are using another IDE, WebSphere Application
Server (with the exception of the Express configuration) provides the Application Assembly Tool (AAT) for packaging applications.

**Application deployment**

Applications are installed on application servers using the administrative console or the wsadmin scripting interface. An application can be deployed to a single server or a cluster. In the case of a cluster, it is installed on each application server in the cluster.

Installing an application involves the following:

- Binding resource references (created during packaging) to actual resources. For example, a data source would need to be bound to a real database.
- Defining JNDI names for EJB home objects.
- Specifying data source entries for entity beans.
- Binding EJB references to the actual EJB JNDI names.
- Mapping Web modules to virtual hosts.
- Specifying listener ports for message-driven beans.
- Mapping application modules to application servers.
- Mapping security roles to users or groups.

After a new application is deployed, the Web server plug-in configuration file needs to be regenerated and copied to the Web server.
IBM WebSphere Extended Deployment

This paper provides a technical overview of WebSphere Extended Deployment software. It describes the core function of the product that can help you derive the most value from your complex computing environment. It also explains how businesses of all sizes, with wide-ranging needs, can leverage WebSphere Extended Deployment software to increase business flexibility and reduce IT complexity.

Note that this chapter can be downloaded as a white paper from:
3.1 Introduction

In today's on demand business environment, your organization has to maintain — and continually improve — the quality of service that it provides, while minimizing IT expenditures. You have to respond quickly to customer and market demand. And your IT infrastructure must be able to keep pace as your business needs change — with a minimum of human intervention. By tightening the link between the technologies that run your business and your overall business goals, you can use your IT infrastructure to help you implement your business strategies more efficiently.

Building robust function into your IT infrastructure can help you reduce IT management complexities to make better use of your existing assets. Improve your organization's ability to adapt to changing customer and trading partner demands on the fly — and be prepared for future growth. Incorporating sense-and-respond capabilities into your infrastructure can increase efficiencies and let you shift valuable IT and human resources to higher value work.

IBM has a long history of providing leading-edge middleware that addresses high availability and reliability — without sacrificing the consistent, predictable performance you need to maintain competitive advantage. IBM WebSphere® Extended Deployment, Version 5.1 delivers on this promise — by adding to the already robust enterprise-class capabilities of IBM WebSphere Application Server Network Deployment, Version 5.1. With its enhanced function, WebSphere Extended Deployment can help you maximize the value to your organization's on demand operating environment.

An on demand operating environment spans integration, virtualization and automation. Robust integration capabilities are a core component of the WebSphere portfolio. IBM clients across the globe use IBM WebSphere Business Integration software from IBM to integrate their back end data stores and to provide enterprise services integration using Web services, process choreography and messaging technologies. It is in the virtualization and automation areas that WebSphere Extended Deployment adds its value.

WebSphere Extended Deployment is built on a virtualized infrastructure that extends the traditional concepts of Java 2 Platform, Enterprise Edition (J2EE) resources and applications, as well as their relationships with one another. This new infrastructure facilitates the ability of WebSphere Extended Deployment to automate operations in an optimal and repeatable fashion. With its automation capabilities, WebSphere Extended Deployment can help you reduce your total cost of ownership (TCO) and provide a more stable, predictable and reliable operating environment.
3.2 An adaptable IT infrastructure

WebSphere Extended Deployment provides an IT infrastructure that dynamically and reliably adapts to changing business demands. By extending the capabilities of WebSphere Application Server Network Deployment, WebSphere Extended Deployment can help you optimize the utilization and management of your deployments and enhance the quality of service of your business-critical applications.

As Figure 3-1 on page cxxx illustrates, WebSphere Extended Deployment capabilities can help you handle the IT scalability and performance challenges of on demand operations. Leveraging the principles and concepts of proven IBM systems and years of IBM research and client experience, WebSphere Extended Deployment enables:

► Dynamic operations
  Allow your application environment to scale as needed with the virtualization of WebSphere resources and the use of a goals-directed infrastructure, helping you increase the speed at which your company can adapt to business demands.

► Extended manageability
  Offers simpler and improved management of complex system operations with advanced, meaningful real-time visualization tools and gradual, controlled implementation of autonomic computing capabilities, helping you reduce the cost of managing IT resources.

► High performance computing
  Enhances the quality of service of business-critical applications to support near linear scalability for high end transaction processing, helping you improve customer service levels.
3.3 Dynamic operations

WebSphere Extended Deployment dynamic operations capabilities help increase responsiveness and flexibility. WebSphere Extended Deployment is designed to deliver dynamic operations through two key capabilities - the virtualization of WebSphere environments and the introduction of a goals-directed infrastructure. A virtualized WebSphere environment allows you to grow your solution as business needs dictate through the dynamic allocation of WebSphere resources. WebSphere Extended Deployment implements a virtualized environment by creating pools of resources that can be shared among applications, thereby optimizing utilization and simplifying overall deployment. As resources are needed for expected (or unexpected) spikes in workload demand, application resources can be allocated to where they are needed most. This enables better usage of the computing resources that you already own and might allow you to run more applications on the machines that you already have in place.
3.3.1 Resource sharing

Sharing resources among applications helps to optimize utilization and simplify deployment. WebSphere Extended Deployment redefines the relationship between traditional J2EE constructs. Instead of deploying applications directly onto a server, you can map an application into a resource pool. This application can then be deployed on some subset of servers within that pool according to your configured business goals. The WebSphere Extended Deployment virtualized infrastructure is predicated on two new constructs: node groups (which represent resource pools) and dynamic clusters.

Node groups

In WebSphere Extended Deployment, the relationship between applications and the nodes on which they can be run is expressed as an intermediate construct called a node group. In concrete terms, a node group is nothing more than a set of machines. In a more abstract sense, a node group is a pool of machines with a common set of capabilities and properties (such as connectivity to a given network or the capability to connect to a certain type of database). These characteristics are not explicitly defined; node group attributes are purely implicit in the WebSphere Extended Deployment design.

Within a node group, one or more dynamic clusters are created. The computing power represented by a node group is divided among its member dynamic clusters. This distribution of resources is modified autonomically according to business goals to compensate for changing workload patterns.

Because a node group's set of common capabilities and properties is required by some suite of applications, a node group defines the collection of machines able to run a given application. Because the administrator now understands what is implied by participation in a given node group, he or she can ensure that applications are deployed into node groups where they can be accommodated. The resources in a given node group are dynamically allocated according to load and policy to deliver better resource utilization, leading to cost savings. Implementing virtualization using node groups breaks the tie between application clusters and machines, and enables them to be shared among applications, optimizing resource utilization and simplifying overall deployment.

Dynamic clusters

The process of deploying an application in WebSphere Extended Deployment begins with choosing or defining a node group that satisfies the application's requirements, and continues with the creation of a dynamic cluster. A dynamic cluster is a container construct that extends its static counterpart in WebSphere Application Server Network Deployment - the static cluster. Dynamic clusters are associated with a single node group. Cluster applications (.ear files) are then
deployed into a dynamic cluster in much the same fashion as they are deployed into WebSphere Application Server Network Deployment static clusters. However, WebSphere Extended Deployment supports autonomic expansion and contraction of a dynamic cluster within its parent node group. Thus, periodic spikes in demand for an application result in a corresponding increase in that application's resource for processing requests. The strategy for increasing these resources is dictated by operational policies that reflect business goals.

### 3.3.2 Application service levels

With WebSphere Extended Deployment, you can differentiate application service levels according to your business requirements. The goals-directed infrastructure capabilities of WebSphere Extended Deployment mean that user requests are classified, prioritized, queued and routed to servers based on application operational policies that are tied to business goals. Application performance is optimized according to these policies that reflect service level goals and relative importance to the organization. Simply put, you can state what applications are important to you, and these applications will get the highest priority access to your WebSphere resources at the right time. This can help you ensure, for example, that your business-critical transactions get the best quality of service.

*Operational policy* is an explicit configurable edict through which operating goals and guidelines are fed into a WebSphere environment. Through operational policy, WebSphere Extended Deployment introduces the capability to designate the business importance of different request types. Among other things, this can help balance your resources optimally in a controlled manner during periods of overutilization. WebSphere Extended Deployment support of operational policy is manifested in two new constructs: *transaction classes* and *service policies*.

#### Transaction classes and service policies

*Transaction classes* define logical request types. They are manifested as lists of Uniform Resource Identifier (URI) filters. For example, administrators can define a transaction class entitled *purchase*, which represents all the buy actions in an enterprise. An administrator assigns a business-importance to a transaction class by mapping it into a service policy.

A *service policy* consists of a performance goal, and a business-importance consists of a performance goal and a business-importance assessment. In times of low demand, operational policy is of little interest, because all requests can be serviced immediately. Performance goals become useful and interesting when demand exceeds computing power. When this occurs, service policies of low importance begin to miss their performance goals, while mission-critical requests maintain a high quality of service, thus achieving graceful and controlled degradation. As shown in Figure 3-2 on page cxxxiii, four service policies have
been defined - platinum, gold, silver and bronze - with corresponding service levels of highest, high, medium and low. The response time goals for these service policies have also been set based on their relative importance to the organization.

The categorization of requests into service policies could be accomplished by mapping URIs directly into service policies. However, this would introduce problems in the areas of reporting and visualization, because statistics would only be available in the scope of a service policy or URI. The transaction class offers a convenient way to facilitate reporting statistics, because it represents some logical grouping of requests that is specific to each enterprise. In this respect, WebSphere Extended Deployment offers customizable visualization and reporting of performance statistics. Classification of incoming requests (into transaction classes) is performed in the on demand router, as is queuing of requests according to service policy. Later in this white paper, the section, The on demand router component, goes into more detail on these processes.

3.3.3 Performance requirements

With WebSphere Extended Deployment, you can quantify performance requirements to help ensure adequate resource provisioning. WebSphere Extended Deployment currently supports two types of performance goals: average response time and discretionary response time. An average response time goal is fairly self-explanatory. It requires a specified time quantity that represents the highest average response time. When the computed average response time for an application exceeds that of its service policy, autonomic
managers take action to mitigate the problem. Discretionary performance goals simply command a best effort from the WebSphere Extended Deployment runtime to service incoming requests. No absolute response time goal is associated with this type of performance goal, however.

**Performance goals as an early warning mechanism**

Many unfavorable circumstances, including overutilization, constrained computing resources and error conditions, can impact your ability to meet performance goals. When response time increases, actions must be taken to correct the situation. In this respect, WebSphere Extended Deployment operational policy facilities can act as an early warning mechanism. If there are problems with reaching performance goals and an administrator knows that incoming traffic is not high enough to cause those problems, he or she is compelled to investigate error conditions, memory leaks or other unfavorable circumstances as potential causes.

### 3.3.4 Focus on business strategy - not the technology

Earlier, this paper introduced the WebSphere Extended Deployment virtualized infrastructure as a basis for its on demand capabilities. This section discusses the capability enabled by that infrastructure: automated enterprise orchestration, or *automation*. Automation brings with it a host of new topological entities, including the on demand router component and several WebSphere Extended Deployment autonomic managers. IBM Tivoli® Intelligent Orchestrator and IBM Tivoli Provisioning Manager (optional, available separately) can be used in enterprise-wide scenarios. The on demand router component controls the flow of work into a WebSphere Extended Deployment topology, the autonomic managers perform autonomic deployment of applications to server instances, and Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager handle the provisioning of resources between resource pools.

**The on demand router component**

The *on demand router* is a component that sits in the front of the WebSphere Extended Deployment implementation. It represents the entry point into a WebSphere Extended Deployment topology and controls the flow of requests into the back end. Specifically, the on demand router handles the queuing and dispatching of requests according to operational policy. In optimizing queue lengths and dispatch rates, several factors are considered, including request concurrency (per node group), operational policy, service policy weights and load balancing. Figure 3-3 on page cxxxv illustrates the on demand router architecture.
Classification of requests
The purpose of the on demand router is to accept incoming requests and to distribute these requests to the WebSphere Extended Deployment back end in an intelligent manner that reflects configured business goals. This process is dependent on the characterization of requests so that each request’s relative business importance can be compared. The characterization begins with the classification of requests into a finite set of request types, or transaction classes. As explained in the section of this white paper entitled Differentiate application service levels according to your business requirements, transaction classes correspond to a list of URIs. Thus, classification of requests is equal to pattern matching on the set of these URI lists.

Class-based queuing of work
As soon as a request has been mapped to a transaction class, it is linked to that transaction class’s service policy. The request is then placed into the queue that corresponds to its service policy; each service policy is assigned a separate queue. An autonomic manager occasionally adjusts dispatching weights of each queue to achieve business goals based on measurements of arrival rates and service times.
Limiting concurrency
To protect against the prospect of an overloaded enterprise, the on demand router limits the number of requests being serviced concurrently in the WebSphere Extended Deployment back end. Concurrency limits are computed based on configured maximum thread-pool sizes. When these limits are reached, requests begin queuing up in the on demand router until demand subsides. If queues reach a (configurable) maximum length, subsequent requests can result in a message returned to the client indicating that the server is too busy to process the incoming request.

3.3.5 Autonomic capabilities
With WebSphere Extended Deployment, you can introduce autonomic capabilities into your infrastructure at your own pace. WebSphere Extended Deployment autonomic capabilities are delivered in a set of components known as autonomic managers. These components monitor performance and health statistics through a series of sensors, and turn various internal control knobs to optimize system performance.

This section takes a look at each of the autonomic managers in a WebSphere Extended Deployment topology.

Flow control manager
The flow control manager oversees the dispatch of requests from the set of queues in the on demand router. It uses a weighted round-robin scheduling algorithm. The flow control manager modifies the weights of queues to align flow control with business goals. After work has been dispatched by the flow control manager, it is passed to the dynamic workload manager.

Dynamic workload manager
The dynamic workload manager handles load balancing of work across an enterprise back end by maintaining a table of servers to which it is delivering work. In this table, each server is dynamically assigned a weight corresponding to its relative capacity to perform work. In WebSphere Extended Deployment terminology, the dynamic workload manager maintains a list of active server instances for each dynamic cluster, and assigns each a routing weight according to observed performance trends. Requests are then routed to candidate server instances to balance workloads on the nodes within a dynamic cluster.

Dynamic cluster managers
Dynamic cluster managers exist in each node agent process in a WebSphere Extended Deployment topology, with only one instance being active in a node group at any given time. The primary responsibility of this autonomic manager is
to control an application's footprint within a node group. Specifically, dynamic cluster managers dictate the location and cardinality of active instances in a given dynamic cluster. As demand increases or decreases, a given dynamic cluster is expanded or contracted, according to operational policy. A dynamic cluster can be deployed on any subset of the nodes in a node group except an empty set.

### 3.3.6 Scale beyond your defined environment

When demand distribution within a WebSphere Extended Deployment enterprise shifts between node groups, the ability to compensate extends beyond the boundaries of WebSphere Extended Deployment autonomic managers. By exploiting Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager, a WebSphere Extended Deployment topology can acquire more physical computing resources from outside a node group.

Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager are IBM's standard tools for provisioning. In general, Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager monitor and manage a set of disparate resource pools (such as WebSphere Application Server, IBM WebSphere MQ, SAP, or e-mail) to help ensure processing power (in the form of machines, logical partitions (LPARs), or processors) is allocated according to business goals. In the specific application, the WebSphere Extended Deployment provisioning engine, Tivoli Intelligent Orchestrator and Tivoli Provisioning Manager are provided with information that they can use to compute optimum allocations of available resources within the scope of a WebSphere Extended Deployment topology. They then modify distributions of resources among candidate workloads.

### 3.4 Extended manageability

WebSphere Extended Deployment's extended manageability capabilities help simplify IT management while maintaining administrator control. It can be difficult to visualize and manage complex IT environments where tens of applications are deployed on hundreds of application servers. Although the WebSphere Application Server administration console provides excellent built-in capabilities, the special needs of very complex deployments require an aggregated, meaningful view of the application run-time environment. WebSphere Extended Deployment extends the administration console to allow operators to see at a glance what is happening in their infrastructures and the relative health of the components. It also enhances the existing WebSphere Application Server administrative console by charting application performance against business goals. Alerts that notify you when intervention is required to deliver on business goals help decrease human-intensive monitoring and management.
3.4.1 View your infrastructure run-time status

To maintain ease-of-use in a product that lends itself to the management of complex deployments in an optionally fully automated fashion, WebSphere Extended Deployment provides enhanced manageability features. These features include a visual console that provides a graphical representation of a dynamic WebSphere Extended Deployment topology, reporting of performance statistics and implementation of administrative operations. These visualization functions are delivered as an extension to the WebSphere Application Server administrative console.

The WebSphere Extended Deployment administrative console contains several tools that help you visualize the inner workings of a WebSphere Extended Deployment topology, so you can remain well-informed about the activities taking place within your environment. Operational views represent an intuitive, central distribution point of information pertaining to health, performance and, potentially, autonomic decisions.

Run-time topology
One such view is the run-time topology view, which is a depiction of the view, or the momentary state of a WebSphere Extended Deployment environment (see Figure 3-4 on page cxxxix). This view refreshes on a configurable interval to provide updated information. The run-time topology contains many useful bits of information, including:

- Application-provisioning activity
- Deployment of dynamic cluster instances
- Processor usage (per node)
- Node-to-node group memberships
- Dynamic cluster-to-node group memberships
- Dynamic workload management weight (per application-server instance)
- Process identification (per application-server instance)
Charting
WebSphere Extended Deployment charting presents administrators with customizable graphs of run-time data observed throughout a WebSphere Extended Deployment environment (see Figure 3-5 on page cxl). This view refreshes on the same configurable interval as the run-time topology view. With WebSphere Extended Deployment, you can chart a wide variety of statistics in six styles of graphs. Supported statistics include:

- Average response time
- Concurrent requests
- Average throughput
- Average queue wait time
- Average service time
- Average queue length
- Average drop rate
Charts can be constructed from several perspectives as well, which gives you flexibility in the scope of the statistics you observe. WebSphere Extended Deployment supports charting from cell, node group, dynamic cluster, service policy, transaction class, J2EE module, and proxy (on demand router) perspectives.

The WebSphere Extended Deployment charting facility also offers a brief historical log of statistical data. As new data points are added to the right side of a chart, old data is displayed until it scrolls off the left side of the chart.

Of course, the length of the history visible on the chart at any point in time depends on the number of data points per sampling period, and the type of chart being viewed.

**Run-time map**
WebSphere Extended Deployment provides an innovative visualization technique for displaying hierarchical data called a *treemap*. It is simply a rectangle that is recursively subdivided into smaller rectangles, each of which represents the collection of nodes at some level in the tree of data being depicted. The significance of each rectangle's size and color is purpose-specific.

As Figure 3-6 on page cxli demonstrates from a WebSphere Extended Deployment perspective, the top-level rectangle represents a cell. This rectangle is subdivided into rectangles that represent node groups in that cell. Each node group rectangle is subdivided into rectangles representing dynamic clusters, and
so on. The size and color of rectangles correlate to the magnitude of some statistic, depending on which treemap is being viewed. Color is typically used to represent health or goal attainment, whereas size typically represents a quantity, such as concurrent requests, number of server instances and so on. By default, the treemap displays the entire WebSphere Extended Deployment topology. However, you can drill down to more fine-grained scopes if you wish by simply clicking on a nonroot rectangle in the treemap.

Figure 3-6  Treemap

The WebSphere Extended Deployment treemap facility can be of particular interest to administrators of very large topologies, because it is well-suited to depicting large sets of data in a concise manner. For instance, the usage of a set of 1000 applications could be quickly observed by viewing a treemap of dynamic clusters and comparing the sizes of rectangles. You can view goal attainment of a set of many service policies by observing the color of rectangles in a map of service policies.

Run-time maps provide a robust search capability, which allows you to single out a subset of the data in an entire map (such as all application server instances in a single node), highlight the top 10 performing applications based on response time, or select the dynamic clusters that have the five lowest, concurrent request values.
3.4.2 Know when intervention is required

WebSphere Extended Deployment extends the administrative console to notify you of decisions made by autonomic managers. Notifications can represent either planned or unplanned events.

**Planned events**
Planned events are those (expected) events for which the WebSphere Extended Deployment run time has an action plan. An example would be an average response time breaching its configured limit, which might trigger an increased dynamic cluster footprint. Depending on the configured level of automation, these events could be presented to the administrator in one of several ways. If WebSphere Extended Deployment is operating in *on demand mode*, the action plan runs and a simple notification is presented to the administrator. In *supervisory mode*, the administrator is presented with the mode, the administrator is presented with the action plan, and prompted for approval. In *manual mode*, WebSphere Extended mode, WebSphere Extended Deployment presents a plan, and you can either follow or ignore the advice.

Over time, as the administrator grows more familiar with WebSphere Extended Deployment and its behavior, such decisions and corresponding actions can happen automatically. With the three modes of operation provided by WebSphere Extended Deployment, you can introduce autonomic capabilities into your IT infrastructure in a controlled and gradual way.

**Unplanned events**
Events that are not assigned action plans are displayed to the administrator as warnings to let him or her know that something unexpected has happened. It is then up to you to develop a plan to correct the situation, if it is indeed problematic.

3.4.3 View the inner workings of your infrastructure

WebSphere Extended Deployment provides insight into the inner workings of various run-time components. These views facilitate debugging problems in a WebSphere Extended Deployment environment and give you a glimpse of what is happening in your topology. The current set of proposed views can enable the visualization of:

- On demand router queue contents
- Application placement decisions
- Dynamic workload management routing tables and weights
- Statistical data presented in the operational views
3.5 High volume transaction requirements

High performance computing is designed to reliably support high volume transaction requirements. To reliably support ultra-high-end transaction processing requirements within a unified WebSphere environment, WebSphere Extended Deployment provides dynamic application partitioning and repartitioning, high end caching, workload management and autonomic high availability management.

The WebSphere partition facility provides an essential capability required to achieve the next level in performance, scalability and availability in J2EE applications. During the past five years of WebSphere Application Server development, a series of design patterns has emerged that, when properly implemented, has proven to significantly alter the performance playing field. The first design pattern that WebSphere Extended Deployment addresses is the partitioning pattern, which introduces the concept of application-specific partitions implemented using the WebSphere partition facility - and potentially resulting in significant throughput improvements. The following section describes the partitioning pattern and how WebSphere Extended Deployment enables this pattern through the WebSphere partition facility.

3.5.1 Partitioning pattern

The partitioning pattern addresses bottlenecks that occur in high volume online transaction processing (OLTP) applications that intensively read and write data to databases and require the utmost in data consistency and availability. Examples of such systems include trading, banking, reservation, and online auction systems. Today's J2EE servers have been optimized for read-mostly environments like commerce systems, where data-caching capabilities can be used to offload the back end database systems. However, as systems observe increased data write (such as database insert, delete, create and update) ratios, these caching systems start to break down because of the ever-strenuous task of maintaining consistency between the cache and database (see Figure 3-7 on page cxliv). These schemes often quickly reach a point of diminishing returns and are better off sending all traffic back to the database and allowing the database to manage consistency. This strategy frequently leads to large and costly database configurations, often running on large symmetric multiprocessor (SMP) machines. In ultra-high-volume environments, these database servers inevitably become a cost and performance bottleneck.
The partitioning pattern aims to offload the database by enabling the application server tier to act as a buffer. It also makes interactions with the database more productive. There are five key elements to the partitioning pattern. Table 3-1 outlines these elements and describes how they are manifested in WebSphere Extended Deployment.

Table 3-1 Five key elements to the partitioning pattern

<table>
<thead>
<tr>
<th>Element</th>
<th>WebSphere capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>▶ WebSphere partitioning facility</td>
</tr>
<tr>
<td>Partition-aware workload</td>
<td>▶ WebSphere Extended Deployment through the on demand router (for HTTP)</td>
</tr>
<tr>
<td>managment</td>
<td>▶ Enterprise JavaBeans (EJB) client through Internet</td>
</tr>
<tr>
<td></td>
<td>▶ Inter-ORB Protocol (IIOP) routing</td>
</tr>
<tr>
<td></td>
<td>▶ Java Message Service (JMS) through pull model</td>
</tr>
<tr>
<td>Leveraging partitions</td>
<td>▶ WebSphere dynamic caching service</td>
</tr>
<tr>
<td></td>
<td>▶ Java Database Connectivity (JDBC) batching</td>
</tr>
<tr>
<td>Highly available partitions</td>
<td>▶ High availability manager for highly available partitions</td>
</tr>
</tbody>
</table>
The following section describes the elements of the partitioning pattern and how they can be used in WebSphere Extended Deployment to achieve significant improvement in J2EE application performance and scalability.

### 3.5.2 Partitioning

As the name suggests, *partitioning* is the essential element in the partitioning pattern. Although partitioning cannot help you improve performance and availability on its own, it can establish the foundation upon which these benefits can be achieved. The WebSphere partition facility lets you create and manage partitions in WebSphere Extended Deployment. A WebSphere partition facility partition can be described in several ways. In its simplest form, a partition is a list of labels that can be created by applications (or metadata found in declarations within applications) whenever they are required. For example, an application can render a set of abstract partition names to represent categories of items being bid upon during auctions: sporting goods, automobiles, toys and antiques (see Figure 3-8).

Partitions can also be expressed as a set of HTTP request URIs:

- `/stock_app/stocks/ibm/*`
- `/stock_app/stocks/xyz/*`
- `/stock_app/stocks/abc/*`

<table>
<thead>
<tr>
<th>Element</th>
<th>WebSphere capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebalancing partitions</td>
<td>➤ WebSphere partitioning facility management bean</td>
</tr>
</tbody>
</table>
3.5.3 Application partitioning using the WebSphere partition facility

You can programmatically calculate partitions using an application-provided hashing function. For example, an application can apply a hash function to an arbitrary token (such as an HTTP header or cookie) to derive a number corresponding to a partition. When an application uses one of the aforementioned methods to create its partitions (typically at application startup or during a repartition operation), the WebSphere partition facility component assigns each partition to a server process in the WebSphere Application Server cluster.

Partitions then are the foundations of the partitioning pattern. The next section describes what to do with the partitions.

3.5.4 Partition-aware workload management

The first step toward making partitions productive is to associate requests (of potentially varied protocols, such as HTTP, IIOP and messaging) with a partition. In this case, a request is classified as having an association with a partition and the work is delivered to the server process where that specific partition has been assigned. For example, in the case of an online auction, if a bidder is bidding on
an item in the sporting goods category, the bid request is routed to the server process associated with the sporting goods partition.

In WebSphere Extended Deployment, there are several supported ways to route a request to partitions:

- **On demand router for HTTP**
  The on demand router component of WebSphere Extended Deployment routes HTTP requests to that partition with which each component is associated. A deployment policy can be included in a WebSphere Extended Deployment application that specifies partitions and the rules by which HTTP requests are mapped to these partitions.

  At application startup, the WebSphere partition facility determines the partitions and assigns them to active server processes in the WebSphere Application Server cluster. The partitions to server process assignments are sent to the on demand routers where the information is used to classify and route incoming requests to the corresponding partitions.

- **EJB clients for IIOP**
  Remote EJB clients can be generated to be partition-aware. When these EJB call remote methods are running in a WebSphere Application Server cluster, a similar classification process occurs in the EJB stub where workload management and routing occurs. If the method call is associated with a partition, requests to that remote method are routed to the server process corresponding to the partition. This behavior is similar to the manner in which work is routed to stateful session EJB in clustered environments. However, with the WebSphere partition facility, the affinity process can be controlled in a more precise manner.

- **JMS pull model**
  Although the WebSphere partition facility does not include explicit support for messaging protocols, you can create a publish-and-subscribe-based messaging application so that publish-and-subscribe topics correspond to partitions. In this case, a server process would only subscribe to topics corresponding to its partition name. With this approach, a server can pull messages off a queue that corresponds to its partition.

### 3.5.5 Leveraging partitions

Partitions can be leveraged to maximize performance and scalability. The ability to define application partitions and route messages to these partitions can be a very powerful capability if it is properly leveraged.

An application can leverage these capabilities to achieve a new level of performance and scalability. With partitions and routing, you can set up areas
of the server where you know work can be performed that will not occur anywhere else in the WebSphere cluster. The three most popular means of exploiting partitions are caching, batching and singleton services.

- **Caching**
  
  An application can leverage the exclusive nature of a partition by aggressively caching data. For example, in an equity trading system, the system is designed to match buy orders against sell orders and to complete trade transactions.

  Partitioning can be used to divide data by stock ticker symbol, allowing more intelligent routing of trading requests and more aggressive caching on servers.

- **Batching**
  
  In the trading system example, new high bids for a given item, can be queued in local memory and then sent to the server in a batch, all at once. In a high volume environment, such batching can produce significant savings in remote interactions with the database.

- **Singleton services**
  
  A partition can be the point in a WebSphere Application Server cluster where a particular function or service of single cardinality resides. In this case, the service is called a singleton because it is the only place in the cluster where this function is running. Singletons allow applications to be broken up and run across the cluster.

By leveraging partitions (such as caching and batching), there is a potential for significant performance improvements in applications that are typically difficult to scale (such as applications with high write-to-read ratios).

### 3.5.6 Recover from failures quickly

Leveraging partitions can significantly improve performance. However, a partition could represent a single point of failure if preventative steps are not taken. For example, in the case of leveraging a partition for data caching, if a partition fails and there is no backup, a considerable amount of cached data can be lost. The same is true when running a singleton service. If the partition goes down, that service can be lost. For these reasons, the WebSphere partition facility is implemented as a highly available service.

The WebSphere partition facility uses the new high availability manager in WebSphere Application Server, which actively monitors processes, services and partitions within a WebSphere Application Server cluster. Each server process in a WebSphere Application Server cluster maintains an awareness of the availability of the other processes in the cluster. If a server process goes down,
the surviving processes work together to try to recover the process, its services and its partitions. For example, if the server process containing the partition for a given online auction category fails, the surviving servers in the cluster can detect this failure and elect a new server process to be the host of this category. The WebSphere partition facility and the high availability manager component are designed to recover from a failure in a short period of time.

3.5.7 Rebalance partitions

To ensure that a system is running optimally, you sometimes have to move partitions to rebalance the load on servers. For example, if the sporting goods category is experiencing a high volume of requests, it might be useful to move it to a more capable server. The WebSphere partition facility includes a Java Management Extension (JMX) MBean to allow such operations. The result of rebalancing a partition is similar to a server process failure. Rebalancing operations, however, are typically something that can be planned and done under program or operator control.

3.6 Summary

WebSphere Extended Deployment includes add-on features that extend WebSphere Application Server Network Deployment in the areas of scalability, usability and adaptability. These extended capabilities facilitate management of larger and more complex deployments, as well as extending the current limits of transaction volume. You can exploit the new WebSphere Extended Deployment virtualized environment to gain the autonomic computing capabilities you need to carry you into the on demand era - potentially translating to lower TCO for your company. Because computing resources are more intelligently managed, you can take advantage of dissimilar usage patterns between different types of work. As a result, autonomic computing capabilities translate to higher levels of adaptability to changing usage patterns. To help ease your organization into the frontier of on demand business, WebSphere Extended Deployment provides extensive visualization and serviceability through graphical interfaces. These tools can offer you a valuable view into WebSphere Extended Deployment to give you confidence as you increase the level of automation of your enterprise. These views can also report events taking place in a WebSphere Extended Deployment topology, and can detail performance and health data. All of these functions are combined in a central control panel to help you keep tabs on your IT investment, thereby enhancing overall usability.
3.7 For more information

To learn more about IBM WebSphere Extended Deployment, contact your IBM representative or IBM Business Partner, or visit:

WebSphere Application Server V6 technical overview

WebSphere Application Server is the implementation by IBM of the Java 2 Enterprise Edition (J2EE) platform. It conforms to the J2EE 1.4 specification. WebSphere Application Server is available in three unique packages that are designed to meet a wide range of customer requirements. At the heart of each package is a WebSphere Application Server that provides the runtime environment for enterprise applications.

This discussion centers on the runtime server component of the following packaging options of WebSphere Application Server for distributed platforms:

- IBM WebSphere Application Server - Express V6, referred to as Express
- IBM WebSphere Application Server V6, referred to as Base
- IBM WebSphere Application Server Network Deployment V6, referred to as Network Deployment
4.1 Application server configurations

At the heart of each member of the WebSphere Application Server family is an application server. Each family has essentially the same architectural structure. Although the application server structure for Base and Express is identical, there are differences in licensing terms, the development tool that is provided, and platform support. With Base and Express, you are limited to stand-alone application servers. Each stand-alone application server provides a fully functional J2EE 1.4 environment.

Network Deployment has additional elements that allow for more advanced topologies such as workload management, scalability, high availability, and central management of multiple application servers.

4.1.1 Stand-alone server configuration

Express, Base, and Network Deployment all support a single stand-alone server environment. With a stand-alone configuration, each application server acts as a unique entity. An application server runs one or more J2EE applications and provides the services required to run those applications.

Multiple stand-alone application servers can exist on a machine, either through independent installations of the WebSphere Application Server code or through multiple configuration profiles within one installation. However, WebSphere Application Server does not provide for common management or administration for multiple application servers. Stand-alone application servers do not provide workload management or failover capabilities.

Figure 4-1 on page cliii shows an architectural overview of a stand-alone application server.
4.1.2 Distributed server configuration

With Network Deployment, you can build a distributed server configuration, which enables central administration, workload management, and failover. In this environment, you integrate one or more application servers into a cell that is managed by a Deployment Manager. The application servers can reside on the same machine as the Deployment Manager or on multiple separate machines. Administration and management is handled centrally from the administration interfaces via the Deployment Manager.

With this configuration, you can create multiple application servers to run unique sets of applications and then manage those applications from a central location. However, more importantly, you can cluster application servers to allow for workload management and failover capabilities. Applications that you install in the cluster are replicated across the application servers. When one server fails, another server in the cluster continues processing. Workload is distributed.
among Web containers and Enterprise JavaBeans containers in a cluster using a weighted round-robin scheme.

Figure 4-2 illustrates the basic components of an application server in a distributed server environment.

4.2 Application servers, nodes, and cells

Regardless of the configuration, the WebSphere Application Server is organized based on the concept of cells, nodes, and servers. While all of these elements are present in each configuration, cells and nodes do not play an important role until you take advantage of the features provided with Network Deployment.
4.2.1 Application servers

The application server is the primary runtime component in all configurations and is where an application actually executes. All WebSphere Application Server configurations can have one or more application servers. In the Express and Base configurations, each application server functions as a separate entity. There is no workload distribution or common administration among application servers. With Network Deployment, you can build a distributed server environment consisting of multiple application servers maintained from a central administration point. In a distributed server environment, you can cluster application servers for workload distribution.

4.2.2 Nodes, node groups, and node agents

A node is a logical grouping of server processes that are managed by WebSphere and that share common configuration and operational control. A node is associated with one physical installation of WebSphere Application Server. In a stand-alone application server configuration, there is only one node.

With Network Deployment, you can configure multiple nodes that you can manage from one common administration server. In these centralized management configurations, each node has a node agent that works with a Deployment Manager to manage administration processes.

A node group is a new concept introduced with WebSphere Application Server V6. A node group is a grouping of nodes within a cell that have similar capabilities. A node group validates that the node is capable of performing certain functions before allowing those functions. For example, a cluster cannot contain both z/OS nodes and nodes that are not z/OS. In this case, you can define multiple node groups, one for the z/OS nodes and one for nodes other than z/OS. A DefaultNodeGroup is automatically created based on the Deployment Manager platform. This node group contains the Deployment Manager and any new nodes with the same platform type.

4.2.3 Cells

A cell is a grouping of nodes into a single administrative domain. In the Base and Express configurations, a cell contains one node. That node may have multiple servers, but the configuration files for each server are stored and maintained individually.

In a distributed server configuration, a cell can consist of multiple nodes which are all administered from a single point. The configuration and application files for all nodes in the cell are centralized into a cell master configuration repository.
This centralized repository is managed by the Deployment Manager process and synchronized with local copies that are held on each of the nodes.

4.3 Servers

WebSphere Application Server supplies application servers which provide the functions that are required to host applications. It also provides the ability to define external servers to the administration process. Table 4-1 shows which types of servers you can define to the WebSphere Application Server administration tools.

Table 4-1  WebSphere Application Server server support

<table>
<thead>
<tr>
<th></th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application server</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Application server clustering</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>External Web server</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>External generic server</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>WebSphere V5 JMS servers</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

4.3.1 Application servers

Application servers provide the runtime environment for application code. They provide containers and services that specialize in enabling the execution of specific Java application components. Each application server runs in its own Java Virtual Machine (JVM).

4.3.2 Clusters

With Network Deployment, you can use application server clustering to enhance workload distribution. A cluster is a logical collection of application server processes that provides workload balancing and high availability.

Application servers that belong to a cluster are members of that cluster and must all have identical application components deployed on them. Other than the applications configured to run on them, cluster members do not have to share any other configuration data.

For example, one cluster member might be running on a large multi-processor server while another member of that same cluster might be running on a small mobile computer. The server configuration settings for each of these two cluster
members is very different, except in the area of the application components that are assigned to them. In that area of configuration, they are identical.

The members of a cluster can be located on a single node (vertical cluster), across multiple nodes (horizontal cluster), or on a combination of the two.

When you install, update, or delete an application, the updates are automatically distributed to all members in the cluster. In WebSphere Application Server V5, if you updated an application on a cluster, you had to stop the application on every server in the cluster, install the update, and then restart the server. With WebSphere Application Server V6, the Rollout Update option allows you to update and restart the application servers on each node, one node at a time, providing continuous availability of the application.

### 4.3.3 JMS servers (V5)

In WebSphere Application Server V5, JMS servers provide the default messaging support for WebSphere Application Server. For migration purposes, Network Deployment in V6 supports cells that contain both V5 and V6 nodes (the Deployment Manager must be at V6), and by extension, Network Deployment supports existing JMS servers in V5 application servers in the cell.

### 4.3.4 External servers

You can define servers other than WebSphere application servers to the administrative process. You can define:

- **Generic servers**
  A generic server is a server that is managed in the WebSphere administrative domain. However, is not a server that is supplied by the WebSphere Application Server product. The generic server can be any server or process that is necessary to support the application server environment, including a Java server, a C or C++ server or process, a CORBA server, or a Remote Method Invocation server.

- **Web servers**
  Web servers can be defined to the administration process as a Web server node, allowing applications to be associated with one or more defined Web servers.
  
  Web server nodes can be managed or unmanaged. Managed nodes have a node agent on the Web server machine that allows the Deployment Manager to administer the Web server. You can start or stop the Web server from the Deployment Manager, generate the Web server plug-in for the node, and automatically push it to the Web server. You would normally have managed
Web server nodes behind the firewall with the WebSphere Application Server installations.

Unmanaged Web server nodes, as the name implies, are not managed by WebSphere. You would normally find these outside the firewall or in the demilitarized zone. You must manually copy or FTP Web server plug-in files to the Web server. However, if you define the Web server as a node, you can generate custom plug-in files for it.

**Note:** As a special case, if the unmanaged Web server is an IBM HTTP Server, you can administer the Web server from the WebSphere administrative console. Then, you can automatically push the plug-in configuration file to the Web server with the Deployment Manager using HTTP commands to the IBM HTTP Server administration process. This configuration does not require a node agent.

### 4.4 Containers

The J2EE 1.4 specification defines the concept of containers to provide runtime support for applications. There are two types of containers in the application server implementation:

- A Web container, which processes HTTP requests, servlets, and JavaServer Pages (JSPs)
- An EJB container, which processes Enterprise JavaBeans (EJBs)

In addition, there is an application client container that can run on the client machine. Table 4-2 shows the containers that each packaging option supports.

<table>
<thead>
<tr>
<th></th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web container</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EJB container</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Application client container</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### 4.4.1 Web container

The Web container processes servlets, JSP files, and other types of server-side includes. Each application server runtime has one logical Web container, which can be modified, but not created or removed. Each Web container provides the following:
Web container transport chains
Requests are directed to the Web container using the Web container inbound transport chain. The chain consists of a TCP inbound channel that provides the connection to the network, an HTTP inbound channel that serves HTTP 1.0 and 1.1 requests, and a Web container channel over which requests for servlets and JSPs are sent to the Web container for processing.

Servlet processing
When handling servlets, the Web container creates a request object and a response object, and then invokes the servlet service method. The Web container invokes the servlet’s destroy method when appropriate and unloads the servlet, after which the JVM performs garbage collection.

HTML and other static content processing
Requests for HTML and other static content that are directed to the Web container are served by the Web container inbound chain. However, in most cases, using an external Web server and Web server plug-in as a front-end to a Web container is more appropriate for a production environment.

Session management
Support is provided for the javax.servlet.http.HttpSession interface as described in the Servlet application program interface (API) specification.

Web services engine
Web services are provided as a set of APIs in cooperation with the J2EE applications. Web services engines are provided to support Simple Object Access Protocol (SOAP).

Web server plug-ins
Although the Web container can serve static content, a more likely scenario is that you will use an external Web server to receive client requests. The Web server can serve requests that do not require any dynamic content, for example, HTML pages. However, when a request requires dynamic content, such as JSP or servlet processing, it must be forwarded to WebSphere Application Server for handling.

To forward a request, you use a Web server plug-in that is included with the WebSphere Application Server packages for installation on a Web server. You copy an Extensible Markup Language (XML) configuration file, configured on the WebSphere Application Server, to the Web server plug-in directory. The plug-in uses the configuration file to determine whether a request should be handled by the Web server or an application server. When WebSphere Application Server receives a request for an application server, it forwards the request to the appropriate Web container in the application server. The plug-in can use HTTP or HTTPS to transmit the request.
4.4.2 Enterprise JavaBeans container

The Enterprise JavaBeans (EJB) container provides all the runtime services that are needed to deploy and manage enterprise beans. It is a server process that handles requests for both session and entity beans.

The enterprise beans (packaged in EJB modules) installed in an application server do not communicate directly with the server. Instead, the EJB container provides an interface between the enterprise beans and the server. Together, the container and the server provide the enterprise bean runtime environment.

The container provides many low-level services, including threading and transaction support. From an administrative viewpoint, the container manages data storage and retrieval for the contained enterprise beans. A single container can host more than one EJB Java archive (JAR) file.

4.4.3 Application client container

The application client container is a separately installed component on the client's machine. It allows the client to run applications in a J2EE environment that is compatible with EJB.

To launch the application client along with its client container runtime, execute the following command:

`launchClient`
4.5 Application server services

The application server provides services other than the containers as shown in Table 4-3.

*Table 4-3 WebSphere Application Server services*

<table>
<thead>
<tr>
<th>Service</th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE Connector Architecture services</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transaction service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic cache service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Message listener service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Object Request Broker service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Administrative service (Java Management Extensions)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Diagnostic trace service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Debugging service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Name service (Java Naming Directory Interface)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance Monitoring Interface service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security service (JAAS and Java 2 security)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Service Integration Bus service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

In addition to these services, Table 4-4 shows the services that are provided to support the programming model extensions.

*Table 4-4 services that support programming model extensions*

<table>
<thead>
<tr>
<th>Service</th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application profiling service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Compensation service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Internationalization service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Object pool service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Startup beans service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Activity session service</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The sections that follow discuss the WebSphere Application Server services that are listed in Table 4-3 on page clxi.

### 4.5.1 J2EE Connector Architecture services

Connection management for access to enterprise information systems (EIS) in WebSphere Application Server is based on the J2EE Connector Architecture (JCA) specification, also sometimes referred to as J2C. The connection between the enterprise application and the EIS is done through the use of EIS-provided resource adapters, which are plugged into the application server. The architecture specifies the connection management, transaction management, and security contracts that exist between the application server and the EIS.

Within the application server, the Connection Manager pools and manages connections. The Connection Manager administers connections that are obtained through both resource adapters defined by the JCA specification and data sources defined by the JDBC 2.0 Extensions (and later) specification.

### 4.5.2 Transaction service

WebSphere applications use transactions to coordinate multiple updates to resources as one unit of work such that all or none of the updates are made permanent. Transactions are started and ended by applications or the container in which the applications are deployed.

WebSphere Application Server is a transaction manager that supports the coordination of resource managers through the XAResource interface and participates in distributed global transactions with transaction managers that support the CORBA Object Transaction Service protocol (for example, application servers) or the Web Service Atomic Transaction protocol.

WebSphere Application Server also participates in transactions imported through J2EE Connector 1.5 resource adapters. You can also configure WebSphere applications to interact with (or to direct the WebSphere transaction service to interact with) databases, Java Message Service (JMS) queues, and JCA connectors through their local transaction support when distributed transaction coordination is not required.
How applications use transactions depends on the type of application component, for example:

- A session bean can either use container-managed transactions (where the bean delegates management of transactions to the container) or bean-managed transactions (where the bean manages transactions itself).
- Entity beans use container-managed transactions.
- Web components (servlets) use bean-managed transactions.

WebSphere Application Server handles transactions with three main components:

- A transaction manager that supports the enlistment of recoverable XAResources and ensures that each such resource is driven to a consistent outcome, either at the end of a transaction, or after a failure and restart of the application server.
- A container in which the J2EE application runs. The container manages the enlistment of XAResources on behalf of the application when the application performs updates to transactional resource managers (for example, databases). Optionally, the container can control the demarcation of transactions for enterprise beans that are configured for container-managed transactions.
- An API (UserTransaction) that is available to bean-managed enterprise beans and servlets which allows such application components to control the demarcation of their own transactions.

### 4.5.3 Dynamic cache service

The dynamic cache service improves performance by caching the output of servlets, commands, Web services, and JSP files. The dynamic cache works within an application server, intercepting calls to objects that can be cached (for example, through a servlet's service() method or a command's execute() method). The dynamic cache either stores the object's output to or serves the object's content from the dynamic cache.

Because J2EE applications have high read-write ratios and can tolerate small degrees of latency in the currency of their data, the dynamic cache can create significant gains in server response time, throughput, and scalability.

The following caching features are available in WebSphere Application Server:

- Cache replication
  
  Cache replication among cluster members takes place using the WebSphere data replication service. Data is generated one time and then copied or replicated to other servers in the cluster, saving execution time and resources.
Cache disk offload
By default, when the number of cache entries reaches the configured limit for a given WebSphere server, eviction of cache entries occurs, allowing new entries to enter the cache service. The dynamic cache includes an alternative feature named disk offload, which copies the evicted cache entries to disk for potential future access.

Edge Side Include caching
The Web server plug-in contains a built-in Edge Side Include (ESI) processor. The ESI processor caches whole pages, as well as fragments, providing a higher cache hit ratio. The cache implemented by the ESI processor is an in-memory cache, not a disk cache. Therefore, the cache entries are not saved when the Web server is restarted.

External caching
The dynamic cache controls caches outside of the application server, such as that provided by the Edge components, an IBM HTTP Server’s FRCA cache that is not z/OS, and a WebSphere HTTP Server plug-in ESI Fragment Processor that is not z/OS. When external cache groups are defined, the dynamic cache matches external cache entries with those groups and pushes out cache entries and invalidations to those groups. This external caching allows WebSphere to manage dynamic content beyond the application server. The content can then be served from the external cache, instead of the application server, improving performance.

4.5.4 Message listener service
With EJB 2.1, an ActivationSpec is used to connect message-driven beans to destinations. However, you can deploy existing EJB 2.0 message-driven beans against a listener port as in WebSphere Application Server V5. For those message-driven beans, the message listener service provides a listener manager that controls and monitors one or more JMS listeners. Each listener monitors a JMS destination on behalf of a deployed message-driven bean.

4.5.5 Object Request Broker service
An Object Request Broker (ORB) manages the interaction between clients and servers, using Internet Inter-ORB Protocol (IIOP). The ORB service enables clients to make requests and receive responses from servers in a network-distributed environment.

The ORB service provides a framework for clients to locate objects in the network and call operations on those objects as though the remote objects were located in the same running process as the client. The ORB service provides
location transparency. The client calls an operation on a local object, known as a stub. Then the stub forwards the request to the desired remote object, where the operation is run, and the results are returned to the client.

The client-side ORB is responsible for creating an IIOP request that contains the operation and any required parameters, and for sending the request in the network. The server-side ORB receives the IIOP request, locates the target object, invokes the requested operation, and returns the results to the client. The client-side ORB demarshals the returned results and passes the result to the stub, which returns the result to the client application, as though the operation had been run locally.

WebSphere Application Server uses an ORB to manage communication between client applications and server applications as well as communication among product components.

### 4.5.6 Administrative service

The administrative service runs within each server JVM. In Base and Express, the administrative service runs in the application server. In Network Deployment, each of the following hosts an administrative service:

- Deployment manager
- Node agent
- Application server

The administrative service provides the necessary functions to manipulate configuration data for the server and its components. The configuration is stored in a repository in the server's file system.

The administrative service has a security control and filtering functionality that provides different levels of administration to certain users or groups using the following administrative roles:

- Administrator
- Monitor
- Configurator
- Operator

### 4.5.7 Name service

Each application server hosts a name service that provides a Java Naming and Directory Interface (JNDI) name space. The service is used to register resources hosted by the application server. The JNDI implementation in WebSphere Application Server is built on top of a Common Object Request Broker Architecture (CORBA) naming service (CosNaming).
JNDI provides the client-side access to naming and presents the programming model that application developers use. CosNaming provides the server-side implementation and is where the name space is actually stored. JNDI essentially provides a client-side wrapper of the name space stored in CosNaming and interacts with the CosNaming server on behalf of the client.

The naming architecture is used by clients of WebSphere applications to obtain references to objects related to those applications. These objects are bound into a mostly hierarchical structure, referred to as a *name space*. The name space structure consists of a set of name bindings, each containing a name relative to a specific context and the object bound with that name. The name space can be accessed and manipulated through a name server.

The following are features of a WebSphere Application Server name space:

- **The name space is distributed**
  
  For additional scalability, the name space for a cell is distributed among the various servers. The Deployment Manager, node agent, and application server processes all host a name server.
  
  The default initial context for a server is its server root. System artifacts, such as EJB homes and resources, are bound to the server root of the server with which they are associated.

- **Transient and persistent partitions**
  
  The name space is partitioned into transient areas and persistent areas. Server roots are transient. System-bound artifacts such as EJB homes and resources are bound under server roots. There is a cell persistent root that is used for cell-scoped persistent bindings and a node persistent root that is used to bind objects with a node scope.

- **Federated name space structure**
  
  A name space is a collection of all names bound to a particular name server. A name space can contain naming context bindings to contexts located in other servers. If this is the case, the name space is said to be a *federated name space*, because it is a collection of name spaces from multiple servers. The name spaces link together to cooperatively form a single logical name space.

  In a federated name space, the real location of each context is transparent to client applications. Clients have no knowledge that multiple name servers are handling resolution requests for a particular requested object.

  In a Network Deployment distributed server configuration, the name space for the cell is federated among the Deployment Manager, node agents, and application servers of the cell. Each such server hosts a name server. All name servers provide the same logical view of the cell name space, with the
various server roots and persistent partitions of the name space being interconnected by means of the single logical name space.

- **Configured bindings**
  You can use the configuration graphical interface and script interfaces to configure bindings in various root contexts within the name space. These bindings are read-only and are bound by the system at server startup.

- **Support for CORBA Interoperable Naming Service (INS) object Uniform Resource Locator (URL)**
  WebSphere Application Server contains support for CORBA object URLs (corbaloc and corbaname) as JNDI provider URLs and lookup names.

Figure 4-3 summarizes the naming architecture and its components.

![Figure 4-3 Naming topology](image)

### 4.5.8 Performance Monitoring Infrastructure service

WebSphere Application Server collects data on runtime and applications through the Performance Monitoring Infrastructure (PMI). This infrastructure is compatible with and extends the JSR-077 specification.
PMI uses a client-server architecture. The server collects performance data from various WebSphere Application Server components and stores it in memory. This data consists of counters such as servlet response time and data connection pool usage. The data can then be retrieved using a Web client, Java client, or Java Management Extensions (JMX) client. WebSphere Application Server contains Tivoli Performance Viewer, which is integrated in the WebSphere administrative console and displays and monitors performance data.

WebSphere Application Server also collects data by timing requests as they travel through the product components. PMI request metrics log the time spent in major components, such as Web containers, EJB containers, and databases. These data points are recorded in logs and can be written to Application Response Time agents that Tivoli monitoring tools use.

4.5.9 Security service

Each application server JVM hosts a security service that uses the security settings held in the configuration repository to provide authentication and authorization functionality.

4.6 Data Replication Service

The Data Replication Service (DRS) is responsible for replicating in-memory data among WebSphere processes. You can use DRS for:

- HTTP session persistence and failover
- Stateful session EJB persistence and failover (new in V6.0)
- Dynamic cache replication

Replication is done through the use of replication domains that consist of server or cluster members that have a need to share internal data. Multiple domains can be used, each for a specific task among a set of servers or clusters. While HTTP session replication and EJB state replication can (and should) share a domain, you need a separate domain for dynamic cache replication.

You can define a domain so that each domain member has a single replicator that sends data to another member of the domain, or so that each member has multiple replicators that send data to multiple members of the domain.

WebSphere Application Server offers two topologies when setting up data replication among servers:

- Peer-to-peer topology
  
  Each application server stores sessions in its own memory and retrieves sessions from other application servers. In other words, each application
server acts as a client by retrieving sessions from other application servers, and each application server acts as a server by providing sessions to other application servers. This mode, working in conjunction with the workload manager, provides hot failover capabilities.

- Client/server topology

Application servers act as either a replication client or a server. Those that act as replication servers store sessions in their own memory and provide session information to clients. They are dedicated replication servers that just store sessions but do not respond to user requests. Client application servers send session information to the replication servers and retrieve sessions from the servers. They respond to user requests and store only the sessions of the users with whom they interact.

### 4.7 Virtual hosts

A virtual host is a configuration that enables a single host machine to resemble multiple host machines. This configuration allows a single physical machine to support several independently configured and administered applications. A virtual host is not associated with a particular node. It is a configuration, rather than a live object, which is why you can create it but you cannot start or stop it.

Each virtual host has a logical name and a list of one or more Domain Name Server (DNS) aliases by which it is known. A DNS alias is the TCP/IP host name and port number that is used to request the servlet (for example, yourHostName:80). When a servlet request is made, the server name and port number entered into the browser are compared to a list of all known aliases in an effort to locate the correct virtual host and serve the servlet. If no match is found, an HTTP 404 error is returned to the browser.

WebSphere Application Server provides two default virtual hosts:

- **default_host**

  This virtual host is used for accessing most applications. The default settings for **default_host** map to all requests for any alias on ports 80, 9443, and 9080. For example:

  http://localhost:80/snoop
  http://localhost:9080/snoop

- **admin_host**

  This virtual host is configured specifically for accessing the WebSphere Application Server administrative console. Other applications are not
accessible through this virtual host. The default settings for admin_host map to requests on ports 9060 and 9043. For example:

http://localhost:9060/admin

4.8 Session management

In many Web applications, users dynamically collect data as they move through the site based on a series of selections on the pages that they visit. Where the user goes and what the application displays may depend on what the user has chosen previously from the site. To maintain this data, the application stores it in a session.

WebSphere supports three approaches to track sessions:

- Secure Sockets Layer (SSL) session identifiers, where SSL session information is used to track the HTTP session ID.
- Cookies, where the application server session support generates a unique session ID for each user and returns this ID to the user's browser using a cookie. The default name for the session management cookie is JSESSIONID. Using cookies is the most common method of session management.
- URL rewriting.

Session data can be kept in local memory cache, stored externally on a database, or kept in memory and replicated among application servers. Table 4-5 on page clxx shows the session support for each WebSphere Application Server configuration.

<table>
<thead>
<tr>
<th></th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookies</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>URL rewriting</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SSL session identifiers</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>In memory cache</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Session persistence using a database</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Memory-to-memory session persistence</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The Servlet 2.4 specification defines the session scope at the Web application level, meaning that session information can only be accessed by a single Web application. However, there may be times when there is a logical reason for multiple Web applications to share information (for example, sharing a user name). WebSphere Application Server provides an IBM extension to the specification that allows session information to be shared among Web applications within an enterprise application. This option is offered as an extension to the application deployment descriptor. No code change is necessary to enable this option. You specify this option during application assembling.

4.8.1 HTTP Session persistence

Many Web applications use the simplest form of session management, the in-memory local session cache. The local session cache keeps session information in memory, which is local to the machine and WebSphere Application Server where the session information was first created. Local session management does not share user session information with other clustered machines. Users only obtain their session information if they return to the machine and WebSphere Application Server holds their session information about subsequent accesses to the Web site.

Most importantly, local session management lacks a persistent store for the sessions it manages. A server failure takes down not only the WebSphere instances running on the server but also destroys any sessions managed by those instances.

By default, WebSphere Application Server places session objects in memory. However, the administrator has the option of enabling persistent session management. This option instructs WebSphere to place session objects in a persistent store. Using a persistent store allows an application server to recover the user session data on restart or another cluster member after a cluster member in a cluster fails or is shut down. Two options for HTTP session persistence are available:

- **Database**
  
  Session information can be stored in a central session database for session persistence.
  
  In a single-server environment, the session can be persisted when the user's session data must be maintained across a server restart or when the user's session data is too valuable to lose through an unexpected server failure.
  
  In a multi-server environment, the multiple application servers hosting a particular application need to share this database information to maintain session states for the stateful components.

- **Memory-to-memory using data replication services**
In a Network Deployment distributed server environment, WebSphere internal replication enables sessions to be shared among application servers without using a database. Using this method, sessions are stored in the memory of an application server, providing the same functionality as a database for session persistence.

### 4.8.2 Stateful session EJB persistence

With WebSphere Application Server V6, you now have failover capability of stateful session EJBs. This function uses data replication services and interacts with the workload manager component during a failover situation.

### 4.9 Web services

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. WebSphere Application Server can act as both a Web service provider and as a requester. As a provider, it hosts Web services that are published for use by clients. As a requester, it hosts applications that invoke Web services from other locations.

WebSphere Application Server supports SOAP-based Web service hosting and invocation.

<table>
<thead>
<tr>
<th>Table 4-6 WebSphere Application Server server support</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Express and Base</strong></td>
</tr>
<tr>
<td>Web services support</td>
</tr>
<tr>
<td>Private UDDI v3 Registry</td>
</tr>
<tr>
<td>Web Services Gateway</td>
</tr>
<tr>
<td>Enterprise Web services</td>
</tr>
</tbody>
</table>

Web services support includes the following:

- **Web Services Description Language (WSDL)**, an XML-based description language that provides a way to catalog and describe services. WSDL describes the interface of Web services (parameters and result), the binding (SOAP, EJB), and the implementation location.

- **Universal Discovery Description and Integration (UDDI)**, a global platform-independent, open framework that enables businesses to discover each other, define their interaction, and share information in a global registry.
UDDI support in WebSphere Application Server V6 includes UDDI V3 APIs, some UDDI V1 and V2 APIs, UDDI V3 client for Java, and UDDI4J for compatibility with UDDI V2 registries. It also provides a UDDI V3 Registry that is integrated in WebSphere Application Server.

- **SOAP**, a lightweight protocol for exchange of information in a decentralized, distributed environment.
- **XML**, a common language for exchanging information.
- **JAX-RPC (JSR 101)**, the core programming model and bindings for developing and deploying Web services on the Java platform. It is a Java API for XML-based RPC and supports JavaBeans and enterprise beans as Web service providers.
- **Enterprise Web services (JSR 109)**, which adds EJBs and XML deployment descriptors to JSR 101.
- **WS-Security**, the specification that covers a standard set of SOAP extensions that can be used when building secure Web services to provide integrity and confidentiality. It is designed to be open to other security models including PKI, Kerberos, and SSL. WS-Security provides support for multiple security tokens, multiple signature formats, multiple trust domains, and multiple encryption technologies. It includes security token propagation, message integrity, and message confidentiality. The specification is proposed by IBM, Microsoft, and VeriSign for review and evaluation. In the future, WS-Security will replace existing Web services security specifications from IBM and Microsoft, including SOAP Security Extensions (SOAP-SEC), WS-Security and WS-License from Microsoft, as well as security token and encryption documents from IBM.
- **JAXR**, an API that standardizes access to Web services registries from within Java. JAXR 1.0 defines access to ebXML and UDDI V2 registries. WebSphere Application Server provides JAXR level 0 support, meaning that it supports UDDI registries.

  JAXR does not map precisely to UDDI. For a precise API mapping to UDDI V2, IBM provides UDDI4J and IBM Java Client for UDDI V3.

- **The SOAP with Attachments API for Java (SAAJ)**, a standard for sending XML documents over the Internet from the Java platform.
IBM value add: In addition to the requirements of the specifications, IBM has added the following features to its Web services support:

- Custom bindings
  JAX-RPC does not support all XML schema types. Custom bindings allow developers to map Java to XML and XML to Java conversions.

- Support for generic SOAP elements
  In cases where you may want generic mapping, this support allows you to eliminate binding and use the generic SOAPElement type.

- Multi-protocol support
  This feature allows a stateless session EJB as the Web service provider, which provides enhanced performance without changes to the JAX-RPC client.

- Client caching
  In WebSphere Application Server V5, there was support for server side Web service caching for Web services providers running within the application server. In addition to this server side caching, WebSphere Application Server V6 introduces caching for Web services clients running within a V6 application server, including the Web Services Gateway.

4.9.1 Enterprise services (JCA Web services)

Enterprise services offer access over the Internet to applications in a platform-neutral and language-neutral fashion. They offer access to enterprise information systems (EIS) and message queues and can be used in a client/server configuration without the Internet. Enterprise services can access applications and data on a variety of platforms and in a variety of formats.

An enterprise service wraps a software component in a common services interface. The software component is typically a Java class, EJB, or JCA resource adapter for an EIS. In services terminology, this software component is known as the implementation. Enterprise services primarily use WSDL and Web Services Invocation Framework (WSIF) to expose an implementation as a service.

Using the Integrated Edition of WebSphere Studio, you can turn Customer Information Control System (CICS) and Information Management System (IMS) transactions into Web services using JCA.
4.9.2 Web service client

Applications that invoke Web services are known as Web service clients or Web service requestors. An application that acts as a Web service client is deployed to WebSphere Application Server like any other enterprise application. No additional configuration or software is needed for the Web services client to function. Web services clients can also be stand-alone applications.

A Web service client binds to a Web service server to invoke the Web service. The binding is done using a service proxy (or stub), which is generated based on the WSDL document for the Web service. This service proxy contains all the information that is needed to invoke the Web service and is used locally by the clients to access the business service. The binding can also be done dynamically using WSIF.

4.9.3 Web service provider

An application that acts as a Web service is deployed to WebSphere Application Server like any other enterprise application. The Web services are contained in Web modules or EJB modules.

Publishing the Web service to a UDDI registry makes it available to anyone searching for it. Web services can be published to a UDDI registry using the Web Services Explorer provided with Rational Application Developer.

When using Rational Application Developer to package the application for deployment, no additional configuration or software is needed for the Web services client to function. The SOAP servlets are automatically added, and a SOAP admin tool is included in a Web module.

If not, you can use the endptEnabler tool found in the WebSphere bin directory to enable the SOAP services within the Enterprise Application Archive (EAR) file and to add the SOAP admin tool.

4.9.4 Enterprise Web Services

The Enterprise Web Services, based on the JSR 109 specification request, uses JAX-RPC in a J2EE environment that defines the runtime architecture as well as implements and deploys Web services in a generic J2EE server. The specification defines the programming model and architecture for implementing Web services in Java based on JSRs 67, 93, 101, and future JSRs related to Web services standards. You can find the list of JSRs at:

http://www.jcp.org/en/jsr/all
4.9.5 IBM WebSphere UDDI Registry

WebSphere Application Server V6 provides a private UDDI registry that implements V3.0 of the UDDI specification. This registry enables the enterprise to run its own Web services broker within the company or to provide brokering services to the outside world. The UDDI registry installation and management is now integrated in with WebSphere Application Server.

You can access the registry for inquiry and publish through:

- The UDDI SOAP API.
- The UDDI EJB client interface.
- The UDDI user console. You can use this Web-based graphical user interface can to publish and to inquire about UDDI entities. However, it provides only a subset of the UDDI API functions.

Security for the UDDI registry is handled using WebSphere security. To support the use of secure access with the IBM WebSphere UDDI Registry, you need to configure WebSphere to use HTTPS and SSL.

A relational database is used to store registry data.

4.9.6 Web Services Gateway

The Web Services Gateway bridges the gap between Internet and intranet environments during Web service invocations. The gateway builds upon the WSDL and the WSIF for deployment and invocation.

With WebSphere Application Server V6, the Web Services Gateway is fully integrated into the integration service technologies, which provides the runtime. The administration is done directly from the WebSphere administrative console.

The primary function of the Web Services Gateway is to map an existing WSDL-defined Web service (target service) to a new service (gateway service) that is offered by the gateway to others. The gateway thus acts as a proxy. Each target service, whether internal or external, is available at a service integration bus destination.

The role formerly played by filters in the V5 Web Services Gateway is now provided by through JAX-RPC handlers. The use of JAX-RPC handlers provides a standard approach for intercepting and filtering service messages. JAX-RPC handlers interact with messages as they pass in to and out of the service integration bus. Handlers monitor messages at ports and take appropriate action, depending upon the sender and content of each message.
Exposing internal Web services to the outside world

Web services hosted internally and made available through the service integration bus are called *inbound services*. Inbound services are associated with a service destination. Service requests and responses are passed to the service through an endpoint listener and associated inbound port.

From the gateway’s point of view, the inbound service is the target service. To expose the target service for outside consumption, the gateway takes the WSDL file for the inbound service and generates a new WSDL file that can be shared with outside requestors. The interface described in the WSDL is exactly the same. However, the service endpoint is changed to the gateway, which is now the official endpoint for the service client. Figure 4-4 diagrams the configuration for exposing Web services through a gateway.

![Service integration bus diagram](image)

**Figure 4-4** Exposing Web services through a gateway

Externally-hosted Web services

A Web service that is hosted externally and made available through the service integration bus is called an *outbound service*. To configure an externally-hosted service for a bus, you first associate it with a service destination. Then, you configure one or more port destinations, one for each type of binding (for example, SOAP over HTTP or SOAP over JMS) through which service requests and responses are passed to the external service.

From the gateway’s point of view, the outbound service is the target service. Mapping a gateway service to the target service allows internal service
requestors to invoke the service as though it were running on the gateway. Again, a new WSDL is generated by the gateway that shows the same interface but that names the gateway as service provider rather than the real internal server. All requests to the gateway service are rerouted to the actual implementation specified in the original WSDL.

Of course, every client could access external Web services by traditional means, but if you add the gateway as an additional layer in between, clients do not have to change anything if the service implementor changes. This scenario is very similar to that shown in Figure 4-4 on page clxxvii, with the difference that the Web service implementation is located at a site on the Internet.

**UDDI publication and lookup**

The gateway facilitates working with UDDI registries. As you map a service for external consumption using the gateway, you can publish the exported WSDL in the UDDI registry. When the services in the gateway are modified, the UDDI registry is updated with the latest changes.

## 4.10 Service integration bus

The service integration bus provides the communication infrastructure for messaging and service-oriented applications, thus unifying this support into a common component. The service integration bus is JMS provider that is JMS 1.1 compliant for reliable message transport and that has the capability of intermediary logic to adapt message flow intelligently in the network. It also supports the attachment of Web services requestors and providers. Service integration bus capabilities have been fully integrated within WebSphere Application Server, enabling it to take advantage of WebSphere security, administration, performance monitoring, trace capabilities, and problem determination tools.

The service integration bus is often referred to as just a *bus*. When used to host JMS applications, it is also often referred to as a *messaging bus*.

Figure 4-5 illustrates the service integration bus and how it fits into the larger picture of an Enterprise Service Bus.
A service integration bus consists of the following:

- **Bus members**
  Application servers or clusters that have been added to the bus.

- **Messaging engine**
  The application server or cluster component that manages bus resources. When a bus member is defined, a messaging engine is automatically created on the application server or cluster. The messaging engine provides a connection point for clients to produce or from where to consume messages.

An application server has one messaging engine per bus of which it is a member. A cluster has at least one messaging engine per bus and can have more. In this case, the cluster owns the messaging engine(s) and determines on which application server(s) the messaging engine(s) will run.
Destinations
The place within the bus to which applications attach to exchange messages. Destinations can represent Web service endpoints, messaging point-to-point queues, or messaging publish/subscribe topics. Destinations are created on a bus and hosted on a messaging engine.

Message store
Each messaging engine uses a set of tables in a data store (JDBC database) to hold information such as messages, subscription information, and transaction states. Messaging engines can share a database, each using its own set of tables. The message store can be backed by any JDBC database supported by WebSphere Application Server.

4.10.1 Application support
The service integration bus supports the following application attachments:

- Web services
  - Requestors using the JAX-RPC API
  - Providers running in WebSphere Application Server as stateless session beans and servlets (JSR-109)
  - Requestors or providers attaching via SOAP/HTTP or SOAP/JMS

- Messaging applications
  - Inbound messaging using JFAP-TCP/IP (or wrapped in SSL for secure messaging). JFAP is a proprietary format and protocol used for service integration bus messaging providers.
  - MQ application in an MQ network using MQ channel protocol.
  - JMS applications in WebSphere Application Server V5 using WebSphere MQ client protocol.
  - JMS applications in WebSphere Application Server V6.

4.10.2 Service integration bus and messaging
With Express or Base, you typically have one stand-alone server with one messaging engine on one service integration bus. With Network Deployment, however, you have more flexibility by using service integration bus and messaging.
Figure 4-6 illustrates two application servers, each with a messaging engine on a service integration bus.

The following are valid topologies:

- One bus and one messaging engine (application server or cluster).
- One bus with multiple messaging engines.
- Multiple buses within a cell which may or may not be connected to each other.
- Buses connected between cells.
- One application server that is a member of multiple buses and that has one messaging engine per bus.
- A connection between a bus and a WebSphere message queue manager.

When using this type of topology, you should consider the following:

- WebSphere message queue can coexist on the same machine as the WebSphere default messaging provider. (In V5, the embedded JMS server and WebSphere MQ could not coexist on the same machine.)
- A messaging engine cannot participate in a WebSphere MQ cluster.
- You can configure the messaging engine to look like another queue manager to WebSphere MQ.
- WebSphere applications can send messages directly to WebSphere MQ or through the service integration bus.
- You can have multiple connections to WebSphere MQ, but each connection must be to a different queue manager.
- WebSphere Application Server V5 JMS client can connect to V6 destinations. Also, a V6 JMS application can connect to an embedded messaging provider in a V5 server if configured. However, you cannot connect a V5 embedded JMS server to a V6 bus.
**Mediation**
Mediation manipulates a message as it traverses the messaging bus (destination). For example, mediation:

- Transforms the message.
- Reroutes the message.
- Copies and routes the message to additional destinations.
- Interacts with non-messaging resource managers (for example, databases).

You control mediation using a mediation handler list. The list is a collection of Java programs that perform the function of a mediation that are invoked in sequence.

**Clustering**
In a distributed server environment, you can use clustering for high availability and scalability. You can add a cluster as a bus member and achieve the following:

- **High availability**
  One messaging engine is active in the cluster. In the event that the messaging engine or server fails, the messaging engine on a standby server is activated.

- **Scalability**
  A single messaging destination can be partitioned across multiple active messaging engines in the cluster. Messaging order is not preserved.

**Quality of service**
You can define quality of service on a destination basis to determine how messages are (or are not) persisted. You can also specify quality of service within the application.

**Message-driven beans**
With EJB 2.1, message-driven beans (MDB) in the application server that listen to queues and topics are linked to the appropriate destinations on the service integration bus using JCA connectors (ActivationSpec objects). Support is also included for EJB 2.0 MDBs to be deployed against a listener port.

### 4.10.3 Web services and the service integration bus
Through the service integration bus Web services enablement, you can:

- Take an internal service that is available at a service destination and make it available as a Web service.
- Take an external Web service and make it available at a service destination.
Use the Web Services Gateway to map an existing service — either an internal service or an external Web service — to a new Web service that appears to be provided by the gateway.

4.11 Security

Table 4-7 shows the security features that the WebSphere Application Server configurations support.

Table 4-7  WebSphere Application Server security support

<table>
<thead>
<tr>
<th></th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java 2 security</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>J2EE security (role mapping)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JAAS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CSIv2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JACC</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security authentication</td>
<td>LTPA, SWAM</td>
<td>LTPA</td>
</tr>
<tr>
<td>User registry</td>
<td>Local OS, LDAP, custom registry</td>
<td>Local OS, LDAP, custom registry</td>
</tr>
</tbody>
</table>

Figure 4-7 on page clxxxiv presents a general view of the logical layered security architecture model of WebSphere Application Server. The flexibility of that architecture model lies in pluggable modules that you can configure according to the requirements and existing IT resources.
WebSphere Application Server security sits on top of the operating system security and the security features provided by other components, including the Java language. This architecture provides the following layers of security:

- Operating system security protects sensitive WebSphere configuration files and authenticates users when the operating system user registry is used for authentication.
- Standard Java security is provided through the JVM that WebSphere and the Java security classes use.
- The Java 2 Security API provides a means to enforce access control, based on the location of the code and who signed it. Java 2 security guards access to system resources such as file I/O, sockets, and properties. WebSphere global security settings allow you to enable or disable Java 2 security and provide a default set of policies. You can activate or inactivate Java 2 security independently from WebSphere global security.

The current principal of the thread of execution is not considered in the Java 2 security authorization. There are instances where it is useful for the authorization to be based on the principal, rather the code base and the signer.

- The Java Authentication and Authorization Services (JAAS) is a standard Java API that allows the Java 2 security authorization to be extended to the code base on the principal as well as the code base and signers. The JAAS programming model allows the developer to design application authentication in a pluggable fashion, which makes the application independent from the
underlying authentication technology. JAAS does not require Java 2 security to be enabled.

- The Common Secure Interoperability protocol adds additional security features that enable interoperable authentication, delegation and privileges in a CORBA environment. It supports interoperability with the EJB 2.1 specification and can be used with SSL.

- J2EE security uses the security collaborator to enforce security policies based on J2EE and to support J2EE security APIs. WebSphere applications use security APIs to access the security mechanisms and implement security policies. J2EE security guards access to Web resources such as servlets/JSPs and EJB methods based on roles that the application developer defines. Users and groups are assigned to these roles during application deployment.

- Java Contract for Containers (JACC) support allows the use of third-party authorization providers for access decisions. The default JACC provider for WebSphere Application Server is the Tivoli Access Manager that is bundled with Network Deployment. The Tivoli Access Manager client functions are integrated in WebSphere Application Server.

- IBM Java Secure Socket Extension is the SSL implementation that WebSphere Application Server uses. It is a set of Java packages that enable secure Internet communications. It implements a Java version of SSLand Transport Layer Security protocols and includes functionality for data encryption, server authentication, message integrity, and client authentication.

WebSphere Application Server security relies on and enhances all the above mentioned layers. It implements security policies in a unified manner for both Web and EJB resources. WebSphere global security options are defined at the cell level. However, individual servers can override a subset of the security configuration. When using mixed z/OS and distributed nodes, the security domain features are merged.

4.11.1 User registry

The pluggable user registry allows you to configure different databases to store user IDs and passwords that are used for authentication and authorization. Only one single registry can be active at a time. There are three options for user registries:

- Local operating system user registry
  When configured, WebSphere uses the operating system’s users and groups for authentication.
LDAP user registry

An LDAP user registry is often the best solution for large scale Web implementations. Most LDAP servers on the market are well equipped with security mechanisms that you can use to securely communicate with WebSphere Application Server. The flexibility of search parameters that an administrator can set to adapt WebSphere to different LDAP schemas is considerable.

Custom user registry

A custom user registry leaves an open door for any custom implementation of a user registry database. You should use the UserRegistry Java interface that the WebSphere API provides to write a custom registry. You can use this interface to access virtually any relational database, flat files, and so on.

4.11.2 Authentication

Authentication is the process of establishing whether a client is valid in a particular context. A client can be either a user, a machine, or an application. The pluggable authentication module allows you to choose whether WebSphere authenticates the user or accepts the credentials from external authentication mechanisms.

An authentication mechanism in WebSphere typically collaborates closely with a user registry when performing authentication. The authentication mechanism is responsible for creating a credential, which is a WebSphere internal representation of a successfully authenticated client user. Not all credentials are created equal. The abilities of the credential are determined by the configured authentication mechanism.

Although WebSphere provides several authentication mechanisms, only a single active authentication mechanism can be configured at once. The active authentication mechanism is selected when configuring WebSphere global security.

WebSphere provides two authentication mechanisms that differ primarily in the distributed security features each supports:

- Simple WebSphere Authentication Mechanism (SWAM) is intended for simple, non-distributed, single application server type runtime environments. The single application server restriction is because this mechanism does not support forwardable credentials. So, if a servlet or EJB in application server process 1 invokes a remote method on an EJB living in another application server process 2, the identity of the caller identity in process 1 is not transmitted to server process 2. What is transmitted is an unauthenticated
credential, which depending on the security permissions configured on the EJB methods, may cause authorization failures.

Because the Simple WebSphere Authentication Mechanism is intended for a single application server process, single sign-on is not supported.

This type of authentication is suitable for simple environments, software development environments, or other environments that do not require a distributed security solution.

SWAM relies on the session ID and is not as secure as LTPA. Thus, we strongly recommend using SSL with this type of authentication.

- Light Weight Third Party Authentication (LTPA)

LTPA is intended for distributed, multiple application servers and machine environments. It supports forwardable credentials and single sign-on. Lightweight Third Party Authentication is able to support security in a distributed environment through the use of cryptography. This allows it to encrypt, digitally sign, and securely transmit authentication related data and later decrypt and verify the signature.

This type of authentication requires that the configured user registry be a central, shared repository such as LDAP or a Windows domain type registry.

### 4.11.3 Authorization

WebSphere Application Server standard authorization features are as follows:

- Java 2 security architecture, which uses a security policy to specify who is allowed to execute code in the application. Code characteristics, such as a code signature, signer ID, or source server, determine whether the code is granted access to be executed.

- JAAS, which extends the Java 2 approach with role-based access control. Permission to execute a code is granted based on the code characteristics and also on the user running it. JAAS programming models allow the developer to design application authentication in a pluggable fashion, which makes the application independent from the underlying authentication technology.

For each authenticated user, a Subject class is created and a set of Principals is included in the subject to identify that user. Security policies are granted based on possessed principals.

WebSphere Application Server provides an internal authorization mechanism that is used by default. As an alternative, you can define external JACC providers to handle authorization decisions. During application installation, security policy information is stored in the JACC provider server using standard interfaces.
defined by JACC. Subsequent authorization decisions are made using this policy information. An exception is that the WebSphere Application Server default authorization engine makes all administrative security authorization decisions.

4.11.4 Security components

Figure 4-8 shows an overview of the security components that come into play in WebSphere Application Security.

Security server

The security server is a component of WebSphere Application Server that runs in each application server process. If multiple application server instances are executed on a single node, then multiple security servers exist on that node.

The security server component is responsible for managing authentication and for collaborating with the authorization engine and the user registry.
Security collaborators

Security collaborators are application server processes that enforce security constraints that the deployment descriptors specify. These processes communicate with the security server every time authentication and authorization actions are required. The following security collaborators are identified:

- **Web security collaborator**
  Resides in the Web container and provides the following services to the application:
  - Checks authentication
  - Performs authorization according to the constraint specified in the deployment descriptor
  - Logs security tracing information

- **EJB security collaborator**
  Resides in the EJB container and uses Common Secure Interoperability Version 2 (CSIv2) and Secure Authentication Service (SAS) to authenticate Java client requests to enterprise beans. The EJB security collaborator works with the security server to perform the following functions:
  - Checks authorizations according to the specified security constraint
  - Supports communication with local user registry
  - Logs security tracing information
  - Communicates with external ORBs using CSIV2 when a request for a remote bean is issued

4.11.5 Security flows

The following sections outline the general security flow.

**Web browser communication**

When a Web browser sends a request to a WebSphere application, the following interactions occur from a security point of view:

1. The Web user requests a Web resource that is protected by WebSphere Application Server.
2. The Web server receives the request and recognizes that the requested resource is on the application server.
3. Using the Web server plug-in, the Web server redirects the request to the Web security collaborator, which performs user authentication.
4. After successful authentication, the Web request reaches the Web container. The Web security collaborator passes the user’s credentials and the security
information contained in the deployment descriptor to the security server for authorization.

5. Upon subsequent requests, authorization checks are performed either by the Web collaborator or the EJB collaborator, depending on what the user is requesting. User credentials are extracted from the established security context.

**Administrative tasks**

Administrative tasks are issued using either the Web-based administrative console or the wsadmin scripting tool, and the following tasks are executed:

1. The administration client generates a request that reaches the server side ORB and JMX MBeans. The JMX MBeans represent managed resources.
2. The JMX MBeans contact the security server for authentication purposes. JMX beans have dedicated roles assigned and do not use user registry for authentication and authorization.

**Java client communication**

When a Java client interacts with a WebSphere application, the following occurs:

1. A Java client generates a request that reaches the server side ORB.
2. The CSIv2 or IBM SAS interceptor performs authentication on the server side on behalf of the ORB, and sets the security context.
3. The server side ORB passes the request to the EJB container.
4. After submitting a request to the access-protected EJB method, the EJB container passes the request to the EJB collaborator.
5. The EJB collaborator reads the deployment descriptor from the EAR file and reads the user credentials from the security context.
6. Credentials and security information is passed to the security server, which validates user access rights and passes this information back to the collaborator.
7. After receiving a response from the security server, the EJB collaborator authorizes or denies access to the user to the requested resource.

**4.12 Resource providers**

Resource providers define resources that running J2EE applications need. Table 4-8 on page cxci shows the resource provider support of the WebSphere Application Server configuration.
4.12.1 JDBC resources

A data source represents a real-world data source, such as a relational database. When a data source object has been registered with a JNDI naming service, an application can retrieve it from the naming service and use it to make a connection to the data source it represents.

Information about the data source and how to locate it, such as its name, the server on which it resides, its port number, and so on, is stored in the form of properties on the DataSource object. This technique makes an application more portable, because the application does not need to hard code a driver name, which often includes the name of a particular vendor. The technique also makes maintaining the code easier. If, for example, you move the data source to a different server, all you need to do is update the relevant property in the data source. You do not need to touch the code using that data source.

Once a data source has been registered with an application server's JNDI name space, application programmers can use it to make a connection to the data source it represents.

The connection is usually a pooled connection. That is, once the application closes the connection, the connection is returned to a connection pool, rather than being destroyed.

Data source classes and JDBC drivers are implemented by the data source vendor. By configuring a JDBC provider, you are providing information about the set of classes that are used to implement the data source and the database driver. That is, the JDBC provider holds the environment settings for the DataSource object.

<table>
<thead>
<tr>
<th>Table 4-8 WebSphere Application Server resource provider support</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Express and Base</strong></td>
</tr>
<tr>
<td>JDBC provider</td>
</tr>
<tr>
<td>Mail providers (JavaMail)</td>
</tr>
<tr>
<td>JMS providers</td>
</tr>
<tr>
<td>Resource environment providers</td>
</tr>
<tr>
<td>URL providers</td>
</tr>
<tr>
<td>Resource adapters</td>
</tr>
</tbody>
</table>
Data sources
In WebSphere Application Server, connection pooling is provided by two parts, a JCA Connection Manager and a relational resource adapter, as shown in Figure 4-9.

![Figure 4-9 Resource adapter in J2EE connector architecture](image)

The JCA Connection Manager provides the connection pooling, local transaction, and security supports. The relational resource adapter provides the JDBC wrappers and JCA CCI implementation that allow applications using bean-managed persistence, JDBC calls, and container-managed persistence beans to access the database JDBC Driver.

**Version 4 data sources:** WebSphere Version 4.0 provided its own JDBC connection manager to handle connection pooling and JDBC access. This support is included with WebSphere Application Server V5 and V6 to provide support for J2EE 1.2 applications. If an application chooses to use a Version 4 data source, the application will have the same connection behavior as in WebSphere Version 4.0.

### 4.12.2 Mail providers
The JavaMail APIs provide a platform and protocol-independent framework for building Java-based mail client applications. The JavaMail APIs require service...
providers, known in WebSphere as protocol providers, to interact with mail servers that run the pertaining protocols.

A mail provider encapsulates a collection of protocol providers. WebSphere Application Server has a Built-in Mail Provider that encompasses the following protocol providers:

- Simple Mail Transfer Protocol (SMTP)
  A popular transport protocol for sending mail. JavaMail applications can connect to an SMTP server and send mail through it by using this SMTP protocol provider.

- Post Office Protocol (POP3)
  The standard protocol for receiving mail.

- Internet Message Access Protocol (IMAP)
  An alternative protocol to POP3 for receiving mail.

These protocol providers are installed as the default and should be sufficient for most applications. To use other protocols, you must install the appropriate service provider for those protocols.

In addition to service providers, JavaMail requires the JavaBeans Activation Framework (JAF) as the underlying framework to deal with complex data types that are not plain text, such as Multipurpose Internet Mail Extensions (MIME), URL pages, and file attachments.

The JavaMail APIs, the JAF, the service providers, and the protocols are shipped as part of WebSphere Application Server using the following Sun licensed packages:

- mail.jar, which contains the JavaMail APIs as well as the SMTP, IMAP, and POP3 service providers.
- activation.jar, which contains the JAF.

### 4.12.3 JCA resource adapters

The JCA defines a standard architecture for connecting the J2EE platform to heterogeneous EIS (for example, ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language).

The JCA resource adapter is a system-level software driver supplied by EIS vendors or other third-party vendors. It provides the connectivity between J2EE components (an application server or an application client) and an EIS.
To use a resource adapter, you need to install the resource adapter code and create connection factories that use the adapter.

One resource adapter, the WebSphere Relational Resource Adapter, is predefined for handling data access to relational databases. This resource adapter provides data access through JDBC calls to access databases dynamically. It provides connection pooling, local transaction, and security support. The WebSphere persistence manager uses this adapter to access data for container-managed persistence beans.

4.12.4 URL providers

URL providers implement the functionality for a particular URL protocol, such as HTTP, by extending the java.net.URLStreamHandler and java.net.URLConnection classes. It enables communication between the application and a URL resource that is served by that particular protocol.

A URL provider named Default URL Provider is included in the initial WebSphere configuration. This provider uses the URL support that the IBM JDK provides. Any URL resource with protocols based on the Java 2 Standard Edition 1.3.1, such as HTTP, FTP or File, can use the default URL provider.

You can also plug in your own URL providers that implement other protocols that the JDK does not support.

4.12.5 JMS providers

The JMS functionality that WebSphere provides includes support for three types of JMS providers:

- Default messaging provider (service integration bus)
- WebSphere MQ provider
- Generic JMS providers
- V5 default messaging provider (for migration)

There can be more than one JMS provider per node. That is, you can configure a node to make use of any combination (or all) of the default messaging provider, WebSphere MQ JMS provider, and a generic JMS provider concurrently. In addition, WebSphere MQ and the default messaging provider can coexist on the same machine.

The support provided by WebSphere administration tools for configuration of JMS providers differs depending upon the provider. Table 4-9 on page cxciv provides a summary of the support.
Table 4-9  WebSphere administration support for JMS provider configuration

<table>
<thead>
<tr>
<th>Configurable objects</th>
<th>Default messaging provider</th>
<th>WebSphere MQ JMS provider</th>
<th>Generic JMS provider</th>
<th>V5 default messaging, WebSphere JMS provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>Messaging system objects (queues/topics)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>JMS administered objects (JMS connection factory and JMS destination)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Default messaging provider**

The default messaging provider for WebSphere Application Server uses the service integration bus for transport. The default message provider provides point-to-point as well as publish and subscribe functions. Within this provider, you define JMS connection factories and JMS destinations that correspond to service integration bus destinations.

**WebSphere MQ messaging provider**

WebSphere Application Server supports the use of full WebSphere MQ as the JMS provider. The product is tightly integrated with the WebSphere installation, with WebSphere providing the JMS client classes and administration interface, while WebSphere MQ provides the queue-based messaging system.

**Generic messaging providers**

WebSphere Application Server supports the use of generic messaging providers, as long as they implement the ASF component of the JMS 1.0.2 specification. JMS resources for generic messaging providers are not configurable using WebSphere administration.

**V5 default messaging provider**

For backwards compatibility with earlier releases, WebSphere Application Server also includes support for the V5 default messaging provider, which enables you to configure resources for use with V5 embedded messaging. You can also use the V5 default messaging provider with a service integration bus.

**4.12.6 Resource environment providers**

The java:comp/env environment provides a single mechanism by which both JNDI name space objects and local application environment objects can be looked up. WebSphere Application Server provides a number of local environment entries by default.
The J2EE specification also provides a mechanism for defining custom (non-default) environment entries using <resource-env-ref> entries that are defined in an application's standard deployment descriptors. The J2EE specification separates the definition of the resource environment entry from the application by:

- Requiring the application server to provide a mechanism for defining separate administrative objects that encapsulate a resource environment entry. The administrative objects are to be accessible via JNDI in the application server's local name space (java:comp/env).
- Specifying the administrative object's JNDI lookup name and the expected returned object type in <resource-env-ref>.

WebSphere Application Server supports the <resource-env-ref> mechanism by providing administration objects for the following:

- Resource environment provider
  Defines an administrative object that groups together the referenceable, resource environment entry administrative objects and any required custom properties.
- Referenceable
  Defines the class name of the factory class that returns object instances implementing a Java interface.
- Resource environment entry
  Defines the binding target (JNDI name), factory class, and return object type (via the link to the referenceable) of the resource environment entry.

### 4.13 Workload management

Clustering application servers that host Web containers automatically enables plug-in workload management for the application servers and the servlets they host. Routing of servlet requests occurs between the Web server plug-in and the clustered application servers using HTTP or HTTPS as shown in Figure 4-10 on page cxcvii.
This routing is based on weights associated with the cluster members. If all cluster members have identical weights, the plug-in sends equal requests to all members of the cluster, assuming no strong affinity configurations. If the weights are scaled in the range from zero to twenty, the plug-in routes requests to those cluster members with the higher weight value more often.

A rule of thumb formula for determining routing preference would be:

\[
\% \text{ routed to Server1} = \frac{\text{weight1}}{\text{weight1} + \text{weight2} + \ldots + \text{weightn}}
\]

where there are \( n \) cluster members in the cluster.

The Web server plug-in temporarily routes around unavailable cluster members.

Workload management for EJB containers can be performed by configuring the Web container and EJB containers on separate application servers. Multiple application servers with the EJB containers can be clustered, enabling the distribution of EJB requests between the EJB containers as shown in Figure 4-11.
In this configuration, EJB client requests are routed to available EJB containers in a round robin fashion based on assigned server weights. The EJB clients can be servlets operating within a Web container, stand-alone Java programs using RMI/IIOP, or other EJBs.

The server weighted round robin routing policy ensures a distribution based on the set of server weights that have been assigned to the members of a cluster. For example, if all servers in the cluster have the same weight, the expected distribution for the cluster is that all servers receive the same number of requests. If the weights for the servers are not equal, the distribution mechanism sends more requests to the higher weight value servers than the lower weight value servers. The policy ensures the desired distribution, based on the weights assigned to the cluster members. In WebSphere Application Server V6, the balancing mechanism for weighted round robin is enhanced to ensure more balanced routing distribution among servers.

You can also choose to have requests sent to the node on which the client resides as the preferred routing. In this case, only cluster members on that node are chosen (using the round robin weight method). Cluster members on remote nodes are chosen only if a local server is not available.

### 4.14 High availability

With Network Deployment V6, the high availability features are significantly improved. The following is a quick overview of the failover capabilities:

- **HTTP server failover**
  The use of multiple HTTP servers, along with a load balancing product such as provided with the Edge components can be used to provide HTTP Server failover.

- **Web container failover**
  The HTTP server plug-in in the Web server is aware of the configuration of all Web containers and can route around a failed Web container in a cluster. Sessions can be persisted to a database or in-memory using data replication services.

- **EJB container failover**
  Client code and the ORB plug-in can route to the next EJB container in the cluster.

- **Deployment manager and node agent**
  The need for failover in these two components has been reduced. Thus, no built-in failover capability is provided. The loss of the Deployment Manager
only affects configuration. We recommend that you use a process nanny to restart the Node Agent if it fails.

- Critical services failover

Hot standby and peer failover for critical services (such as workload management routing, PMI aggregation, JMS messaging, transaction manager, and so on) is provided through the use of high availability domains.

A high availability domain defines a set of WebSphere processes (a core group) that provides high availability function to each other. Processes in the core group can be the Deployment Manager, node agents, application servers or cluster members.

One or more members of the core group can act as a high availability coordinator, managing the HA activities within the core group processes. If a high availability coordinator server fails, another server in the core group takes over the duties of that coordinator. High availability policies define how the failover occurs.

Workload management information is shared between core members and failover of critical services is done among them in a peer-to-peer fashion. Little configuration is necessary, and in many cases, this function works with the defaults that are created automatically as you create the processes.

- Transaction log hot standby

With V6, transaction logs can be maintained on Network Attached Storage. When a cluster member fails, another cluster member recovers the transaction log, thus enabling the failover 2PC transactions.

- JMS messaging failover

The messaging engine keeps messages in a remote database. When a server in a cluster fails, WebSphere selects an online server to run the Messaging Engine and the workload manager routes JMS connections to that server.

### 4.15 Administration

WebSphere Application Server's administration model is based on the JMX framework. JMX allows you to wrap hardware and software resources in Java and expose them in a distributed environment. JMX also provides a mapping framework for integrating existing management protocols, such as SNMP, into JMX’s own management structures.

Each application server has an administration service that provides the necessary functions to manipulate configuration data for the server and its
components. The configuration is stored in a repository. The repository is actually a set of XML files that are stored in the server's file system.

4.15.1 Administration tools

Table 4-10 shows the administration tools that WebSphere Application Server supports by configuration.

<table>
<thead>
<tr>
<th>Administrative console</th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commands</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Scripting client, wsadmin</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Administrative console

The administrative console is a Web-based interface that provides configuration and operation capability. The administrator connects to the application using a Web browser client. Users assigned to different administration roles can manage the application server and certain components and services using this interface.

The administrative console is a system application, crucial to the operation of WebSphere and, as such, is not exposed as an enterprise application on the console. In stand-alone application servers, the administrative console runs in the application server. In the Network Deployment distributed server environment, the administrative console application runs on the Deployment Manager. When a node is added to a cell, the administrative console application is deleted from the node and the configuration files are integrated into the master cell repository that the Deployment Manager maintains.

Commands

WebSphere Application Server provides a set of commands in the 
<server_install>/bin directory that allows you to perform a subset of administrative functions. For example, you can use the startServer command to start an application server.

Scripting client

The wsadmin scripting client provides extra flexibility over the Web-based administration application, allowing administration to use the command-line interface. Using the scripting client not only makes administration quicker, but it automates the administration of multiple application servers and nodes using scripts.
The scripting client uses the Bean Scripting Framework, which allows you to use a variety of scripting languages for configuration and control. WebSphere Application Server V6 supports two languages: jacl and jython (or jpython).

The wsadmin scripting interface is included in all WebSphere Application Server configurations but is targeted toward advanced users. The use of wsadmin requires in-depth familiarity with application server architecture and a scripting language.

### 4.15.2 Configuration repository

The configuration repository holds copies of the individual component configuration documents stored in XML files. The application server's administrative service takes care of the configuration and makes sure it is consistent during the runtime.

The configuration of unfederated nodes can be archived for export and import, making them portable among different WebSphere Application Server instances.

### 4.15.3 Centralized administration

The Network Deployment package allows multiple servers and nodes to be administered from a central location. This centralized administration uses a central Deployment Manager that handles the administration process and distributes the updated configuration to the node agent for each node. The node agent, in turn, maintains the configuration for the servers in the node. Table 4-11 on page ccii shows the distributed administration that WebSphere Application Server supports by configuration.

All operating system processes that are components of the WebSphere product are called managed servers or managed processes. JMX support is embedded in all managed processes. These processes are available to receive administration commands and to output administration information about the state of the managed resources within the processes.

WebSphere provides the following managed servers and processes:

- **Deployment manager**
  Provides a single point to access configuration information and control for a cell. The Deployment Manager aggregates and communicates with the node agent processes on each node in the system.

- **Node agent**
  Aggregates and controls the WebSphere managed processes on its node. There is one node agent per node.
- **Application server**
  Managed server that hosts J2EE applications.

Table 4-11 shows the managed processes supported by each packaging option.

**Table 4-11  WebSphere Application Server distributed administration support**

<table>
<thead>
<tr>
<th></th>
<th>Express and Base</th>
<th>Network Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment manager</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Node agent</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Application servers</td>
<td>Stand-alone</td>
<td>Stand-alone or distributed server clustering</td>
</tr>
</tbody>
</table>

**Deployment manager**
The Deployment Manager process provides a single, central point of administrative control for all elements in the cell. It hosts the Web-based administrative console application. Administrative tools that need to access any managed resource in a cell usually connect to the Deployment Manager as the central point of control. Using the Deployment Manager, horizontal scaling, vertical scaling, and distributed applications are all easy to administer and manage. Application servers are managed by nodes, and one or more nodes is managed by a cell.

In a distributed server environment, the Deployment Manager maintains a master configuration repository that contains all of the cell’s configuration data. The configuration repository at each node is a synchronized subset of the master repository. The node repositories are read-only for application server access. Only the Deployment Manager can initiate their update and push out configuration changes from the cell master configuration repository. It manages through communication with the node agent process resident on each node of the cell.

**Node agent**
The node agent is an administrative process and is not involved in application serving functions. It hosts important administrative functions such as:

- File transfer services
- Configuration synchronization
- Performance monitoring
The node agent aggregates and controls all the managed processes on its node by communicating with:

- The Deployment Manager to coordinate configuration synchronization and to perform management operations on behalf of the Deployment Manager.
- Application servers and managed Web servers to manage (start or stop) each server and to update its configuration and application binaries as required.

Only one node agent is defined (and run) on each node. In a stand-alone server environment, there is no node agent.

### 4.16 The flow of an application

Figure 4-12 shows the typical application flow for Web browser clients using either JDBC (from a servlet) or EJB to access application databases.

![Application flow diagram](image)

Figure 4-12  Application flow

The typical application flow is as follows:

1. A Web client requests a URL in the browser (input page).
2. The request is routed to the Web server over the Internet.
3. The Web server immediately passes the request to the Web server plug-in. All requests go to the WebSphere plug-in first.
4. The Web server plug-in examines the URL, verifies the list of host name aliases from which it will accept traffic based on the virtual host information, and chooses a server to handle the request.

5. A stream is created. A stream is a connection to the Web container. It is possible to maintain a connection (stream) over a number of requests. The Web container receives the request and, based on the URL, dispatches it to the proper servlet.

6. If the servlet class is not loaded, the dynamic class loader loads the servlet (servlet init(), then doGet() or doPost()).

7. JNDI is used for lookup of either datasources or EJBs required by the servlet.

8. Depending upon whether a datasource is specified or an EJB is requested, the JNDI directs the servlet:
   – To the corresponding database and gets a connection from its connection pool in the case of a data source.
   – To the corresponding EJB container, which then instantiates the EJB when an EJB is requested.

9. If the EJB requested involves an SQL transaction, it goes back to the JNDI to look up the datasource.

10. The SQL statement is executed and the data retrieved is sent back either to the servlet or to the EJB.

11. Data beans are created and handed off to JSPs in the case of EJBs.

12. The servlet sends data to JSPs.

13. The JSP generates the HTML that is sent back through the WebSphere plug-in to the Web server.

14. The Web server sends the output page (output HTML) to the browser.
4.17 Developing and deploying applications

Figure 4-13 shows a high-level view of the stages of application development and deployment.

![Diagram of application development and deployment]

**4.17.1 Application design**

Design tools like Rational Rose or Rational XDE can be used to model the application using the Unified Modeling Language. The output of the modeling generally consists of use-case scenarios, class diagrams, and starter code generated based on the model.

**4.17.2 Application development**

Application development is done using Rational Application Developer (or a comparable IDE) to create the enterprise application. You can start by importing pre-generated code from modeling tools, a sample application, an existing production application, or you can start from scratch.

Rational Application Developer provides many tools and aids to get you started quickly. It also supports team development using CVS or Rational ClearCase, which allows multiple developers to share a single master source copy of the code.
During the development phase, you can do component testing using the built-in WebSphere Application Server test environment. Rational Application Developer provides server tools capable of creating and managing servers both in the test environment and on remote server installations. The application is automatically packaged into an EAR file for deployment when you run the application on a server using Rational Application Developer.

### 4.17.3 Application packaging

J2EE applications are packaged into EAR files to be deployed to one or more application servers. A J2EE application contains any or all of the modules as shown in Table 4-12.

**Table 4-12  J2EE 1.3 application modules**

<table>
<thead>
<tr>
<th>Module</th>
<th>Filename</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web module</td>
<td>&lt;module&gt;.war</td>
<td>Servlets, JSP files, and related code artifacts.</td>
</tr>
<tr>
<td>EJB module</td>
<td>&lt;module&gt;.jar</td>
<td>Enterprise beans and related code artifacts.</td>
</tr>
<tr>
<td>Application client</td>
<td>&lt;module&gt;.jar</td>
<td>Application client code.</td>
</tr>
<tr>
<td>module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resource adapter</td>
<td>&lt;module&gt;.rar</td>
<td>Library implementation code that your application uses to connect to enterprise information systems (EIS).</td>
</tr>
<tr>
<td>module</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This packaging is done automatically in Rational Application Developer when you export an application for deployment. If you are using another IDE, WebSphere Application Server (with the exception of the Express configuration) provides the Application Server Toolkit for packaging applications.

**WebSphere Enhanced EAR files**

The WebSphere Enhanced EAR, introduced in WebSphere Application Server V6, is a regular J2EE EAR file with additional configuration information for resources usually required by J2EE applications. While adding this extra configuration information at packaging time is not mandatory, it can simplify deployment of J2EE applications to WebSphere.

When you deploy an enhanced EAR to a WebSphere Application Server V6 server, WebSphere can configure the resources specified in the enhanced EAR automatically. This automatic configuration reduces the number of steps that are required to set up the WebSphere environment to host the application.
4.17.4 Application deployment

Applications are installed on application servers using the administrative console or the wsadmin scripting interface. You can deploy an application to a single server or a cluster. In the case of a cluster, it is installed on each application server in the cluster.

Installing an application involves the following:

- Binding resource references (created during packaging) to actual resources. For example, a data source would need to be bound to a real database.
- Defining JNDI names for EJB home objects.
- Specifying data source entries for entity beans.
- Binding EJB references to the actual EJB JNDI names.
- Mapping Web modules to virtual hosts.
- Specifying listener ports for message-driven beans.
- Mapping application modules to application servers.
- Mapping security roles to users or groups.

The use of an enhanced EAR file simplifies this installation process.

After a new application is deployed, the Web server plug-in configuration file needs to be regenerated and copied to the Web server.

Application update

In previous releases, deploying an update to an application required a complete EAR file to be deployed and the application to be restarted. WebSphere Application Server V6 allows partial updates to applications and makes it possible to restart only parts of an application.

Updates to an application can consist of individual application files, application modules, zipped files that contain application artifacts, or the complete application. All module types can be started (though only Web modules can be stopped).

In V6, you have a rollout start option for installing applications on a cluster that will stop, update, and start each cluster member in turn, ensuring availability.

4.17.5 WebSphere Rapid Deployment

WebSphere Rapid Deployment is designed to simplify the development and deployment of WebSphere applications. It is a collection of Eclipse plug-ins that can be integrated within development tools or run in a headless mode from a
user file system. WebSphere Rapid Deployment is currently integrated in Rational Web Developer, Rational Application Developer, and the Application Server Toolkit. Initially, there are features that are only supported in headless mode.

During development, annotation-based programming is used. The developer adds metadata tags into the application source code that are used to generate artifacts needed by the code, thus reducing the number of artifacts the developer needs to create.

These applications are packaged into an enhanced EAR file that contains the J2EE EAR file along with deployment information, application resources, and properties (environment variables, JAAS authentication entries, shared libraries, classloader settings, and JDBC resources). During installation, this information is used to create the necessary resources. Moving an application from one server to another also moves the resources.

WebSphere Rapid Deployment automates installation of applications and modules onto a running application server by monitoring the workspace for changes and then driving the deployment process.

4.18 Technology support summary

Table 4-13 highlights the support that each WebSphere Application Server packaging option provides.

<table>
<thead>
<tr>
<th>Client and server support for the Software Development Kit for Java Technology Edition 1.4 (SDK 1.4.2)</th>
<th>Base and Express V6</th>
<th>Network Deployment V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE 1.2, 1.3 programming support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Feature</td>
<td>Base and Express V6</td>
<td>Network Deployment V6</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>J2EE 14. programming support†</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>† EJB 2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† Servlet 2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JSP 2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JMS 1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JTA 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JavaMail 1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JAF 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JAXP 1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† Connector 1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† Web Services 1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JAX-RPC 1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† SAAJ 1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JAXR 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† J2EE Management 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JMX 1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JACC 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JDBC 3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebSphere Rapid Deployment</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Service Data Object (SDO)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Messaging support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>† Integrated JMS 1.1 messaging provider</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† Support for WebSphere MQ and generic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† messaging providers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† Message-driven beans</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web services runtime support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Security support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>† Java 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† J2EE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JACC 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† JAAS 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† CSIV2 and SAS authentication protocols</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† LDAP or local operating system user registry</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† LTPA authentication mechanism</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† Kerberos, Technology Preview</td>
<td></td>
<td></td>
</tr>
<tr>
<td>† Simple WebSphere Authentication Mechanism (SWAM)</td>
<td>Yes</td>
<td>stand-alone server environment only</td>
</tr>
</tbody>
</table>

Chapter 4. WebSphere Application Server V6 technical overview  ccix
<table>
<thead>
<tr>
<th>Feature</th>
<th>Base and Express V6</th>
<th>Network Deployment V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-node management and Edge components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workload management and failover</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Deployment manager</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Central administration of multiple nodes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Load Balancer</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Caching Proxy</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic caching</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance and analysis tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance Monitoring Instrumentation (PMI)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Log Analyzer</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Tivoli Performance Viewer (integrated in the administration console)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Administration and tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Administration and tools</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Web-based administration console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integrated IBM HTTP Server and Application Server Administration Console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Administrative scripting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Java Management Extension (JMX) 1.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J2EE Management (JSR-077)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>J2EE Deployment (JSR-088)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Application Server Toolkit</td>
<td></td>
</tr>
<tr>
<td>Web services</td>
<td>Base and Express V6</td>
<td>Network Deployment V6</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>---------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>JAX-RPC v1.0 for J2EE 1.3, v1.1 for J2EE 1.4</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JSR 109 (Web services for J2EE)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS-I Basic Profile 1.1.2 support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS-I Simple SOAP Binding Profile 1.0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS-I Attachments Profile 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAAJ 1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDDI V2 and V3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JAXR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WS-TX (transactions)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOAP 1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSDL 1.1 for Web services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WSIL 1.0 for Web services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OASIS Web Services Security: UsernameToken Profile 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OASIS Web Services Security X.509 Certificate Token Profile</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Web Services Gateway                                                        | No                   | Yes                   |
| Private UDDI v3 Registry                                                     | Yes                  | Yes                   |

### Programming model extensions

| Activity sessions                                                           | Yes               | Yes               |
| Application Profiling                                                       |                   |                   |
| Asynchronous Beans (now called WorkManager)                                 |                   |                   |
| Dynamic caching                                                             |                   |                   |
| Dynamic query                                                               |                   |                   |
| Internationalization Service                                                |                   |                   |
| Object Pools                                                                |                   |                   |
| Scheduler Service (now called Timer Service)                                |                   |                   |
| Startup Beans                                                               |                   |                   |
| WorkArea Service                                                            |                   |                   |
| Extended JTA Support                                                        |                   |                   |
| Last Participant Support                                                     |                   |                   |
| Back-up Cluster Support                                                     | No                | Yes               |

1. You can see the APIs required for J2EE 1.4 in the Application Programming Interface section of the J2EE 1.4 specifications at: http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf
2. Business process choreography and business rule beans remain in WebSphere Business Integration Server Foundation.
WebSphere Business Integration Server Foundation architecture and overview

WebSphere Business Integration Server Foundation V5.1 extends the capabilities of WebSphere Application Server V5.1 by adding the following:

- Process Choreographer technology
- Programming Model Enhancements (PMEs)
- Common Event Infrastructure technology preview

This paper addresses these features of WebSphere Business Integration Server Foundation.

For information on the architectural features of WebSphere Application Server, see *WebSphere Application Server V5 Architecture*, REDP-3721.

For more details on WebSphere Business Integration Server Foundation, see *WebSphere Business Integration Server Foundation V5.1 Handbook*, SG24-6318.
5.1 Process choreography support

The process choreographer in WebSphere Business Integration Server Foundation provides a way to combine workflow technology with the services offered by J2EE within WebSphere Application Server. In addition to workflow-enabling J2EE applications and components, the intent is to allow for independent Web services to be coordinated along with J2EE applications. Integrating these features within WebSphere means that the many features of WebSphere Application Server, including centralized management, workload management, logging, debug and trace features, are available when executing business processes.

Process choreographer support is provided through a set of APIs and the Business Process Execution (BPE) container, which are installed via the Integration Server package on top of WebSphere Application Server base or Network Deployment. The BPE container is implemented as a J2EE application that uses the underlying WebSphere Application Server runtime services and resources.

5.1.1 Business processes

In our daily activities and businesses, we repeatedly perform sequences of activities to achieve an objective. These well-defined and repeatable patterns of activities can be modeled as processes.

A business process is a set of business-related activities that are invoked in a specific sequence to achieve a business goal. The business process defines the sequence of the flow, how external events are to be handled, human interaction requirements, and conditional processing.

For example, Figure 5-1 on page ccxv shows a business process flow that invokes five activities. activity 1 (a Java class or an EJB) has its output connected to activity 2 (a CICS application) and activity 3 (other process flow). At the end of activity 1, both activities 2 and 3 could be executed in parallel, or only one executed based on the condition specified in the connections. The process flow will wait in Activity 4 for a human decision to go to activity 5.
As you can see, a business process-based application consists of both the business process and the applications it invokes. The business process will execute in a process engine and access business applications running on application servers using service-oriented architecture. This effectively separates the business flow logic from the implementation of each individual function.

**BPEL4WS support:** WebSphere Application Server Enterprise V5.0 provided support for business processes developed in Flow Definition Markup Language (FDML). With WebSphere Business Integration Server Foundation V5.1, support has been introduced for business processes developed in Business Process Execution Language for Web Services (BPEL4WS).

BPEL4WS will replace FDML, and further discussions in this paper about business processes will assume BPEL4WS.
5.1.2 Building business processes with WebSphere Studio

WebSphere Studio Application Developer Integration Edition V5.1 provides the support necessary for developing and testing business processes based on BPEL4WS, primarily through its business integration tools.

The Business integration tools are built to support a services-oriented architecture, which decreases development complexity by representing all objects, components, applications, and processes as services (based on open standards). These services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Internet, automating a wide range of business functions, including existing IT assets. Using this type of architecture, developers can create and interact with all software components without regard to their underlying implementation.

Business Integration perspective
The Business Integration perspective provides intuitive drag-and-drop tools to allow you to visually choreograph the sequence and flow of information between services to implement a business process. Using the Process Editor and wizards, developers can:

- Visually compose a service out of one or more existing services.
- Define and transform the processing of information between services.
- Create a process that has within itself other nested processes, to multiple levels.
- Include activities that require human interaction as steps in an automated business process.
- Develop a process from the top down by creating a skeleton process that choreographs the sequence of events in an application.
- Generate deploy code for enterprise services and business processes.

Server tools, also accessible from the Business Integration perspective, allow you to deploy and test applications to remote servers, or to the integrated test environment provided with WebSphere Studio.

Process Editor
The Process Editor displays process-related programming language graphically. You use this tool to visually create and manipulate business processes. The Process Editor acts as a front-end for a service model that is declared in Web Services Definition Language (WSDL). The easiest way to differentiate between the business process programming language and WSDL is that the former defines what to do, and the latter defines how to do it. When you are finished with your process, it will be modeled as a WSDL service without deployment.
information. In other words, the finished process is abstract in that the interface is defined by port types and operations and not by an actual binding.

In Figure 5-2 you can see the WebSphere Studio workbench viewed using the Business Integration perspective. In the top right window, you can see a business process file (.bpel file) opened with the Process Editor. Note that the process itself is contained within a WebSphere Studio service project.

![Business Integration perspective](image)

**Figure 5-2  Business Integration perspective**

The Process Editor areas are as follows:

- The palette (1)

  The palette is a shaded area containing activity icons to the left of the canvas. Use the objects in the palette to assemble your process on the canvas.
The variables area (2)
This is the area of the canvas where the variables are listed. The directions in which the arrows point indicate how they are being used by the activities. For example, if there is an arrow pointing to the left, then the activity that is currently selected on the canvas stores information in this variable. If the arrow points to the right, then this activity reads from this variable. If there is a double-arrow, then the activity reads from and writes to this variable. Note: This is true for all other activities except the Invoke and Staff activities, where this behavior is reversed. To add a new variable, click (+); or to remove one, click (-).

The activities area (3)
This is the area of the canvas where the activities are positioned. Structured activities (that contain other activities) can be expanded or collapsed by clicking either the miniature plus (+) or minus (-) icon. To add an activity, select it in the palette, and drop it anywhere on the canvas. It will automatically be placed here.

The action bar (4)
The action bar is a miniature dialogue that pops up beside the activity that you have selected and contains a series of one or more icons that are relevant to that activity.

The partner links area (5)
This is the area of the canvas where the partners are listed. To add a new one, click (+), or to remove one, click (-).

The canvas (6)
The canvas is the white area that comprises most of the editor. You build the process here by dropping objects from either the palette or the Services view. These objects are organized into distinct areas on the canvas.

The details area (7)
The details area displays properties that are relevant to the object that is currently selected on the canvas. Click the tabs to the left of this view to toggle through the pages. Some pages display properties in tabular format, and you can add or modify these properties by clicking the appropriate cell and then interacting with the graphical interface that appears.

Note that properties for the business process can be set in the detail area with the icon for the process (8) selected.

The correlation sets area (9)
This is the area of the canvas where the correlation sets are listed. To add a new one, click (+); or to remove one, click (-).
Process Debugger
The Process Debugger provides a dramatic increase in programming productivity by providing the user with a visual means of debugging complex processes. The process debugger:

- Provides functionality to step through the process activity-by-activity, including stepping over and into activities
- Seamlessly interacts with the Java Debugger to debug the underlying implementation
- Includes functionality to change the input and output data of an activity while debugging for unit testing

5.1.3 Building business processes
Let us take a quick look at how a business process is built. This section will take you through the basics required to use WebSphere Studio Application Developer Integration Edition to build a simple business process. For information on using the full range of features and to build more complex business processes, refer to the WebSphere Studio Application Developer Integration Edition Help, or WebSphere Business Integration Server Foundation V5.1 Handbook, SG24-6318.

Create a service project
In WebSphere Studio, a service project is used to contain and manage the development of services and processes. WebSphere Studio provides a wizard to help you build the service project (File → New → Service Project).

Create a business process
A wizard is also provided to create a business process (File → New → Business Process) and place it in the service project. During the wizard, you can choose the type of process to build. The options are flow-based BPEL process, sequence-based BPEL process, or V5.0 process (FDML).
Sequence vs. flow-based processes:

What is the difference between a flow-based and sequence-based BPEL process?

BPEL evolved out of both IBM- and Microsoft-based specifications. Flow-based processes came from the IBM definition where we wire control links between activities. Sequence-based flows came from the Microsoft definition where activities are grouped together in a sequence, so no wiring is used because each activity is invoked sequentially. Both are valid approaches and both are supported in Integration Edition.

Any business process can contain both flow and sequence structures. Selecting to create a flow-based or sequence-based process simply defines the starting point for the process.

The wizard will build the file structure required for the process, including WSDL files to be used to interface with the process and a skeleton .bpel file to hold the process.

Modify the WSDL for the business process

WebSphere Studio provides an editor for updating WSDL files. When you create a business process, two default WSDL files are created:

- `<process_name>.wsdl`
- `<process_name>Interface.wsdl` (imported into `<process_name>.wsdl`)

To add to or modify the WSDL to accommodate input or output parameters required by the process, simply open the `<process_name>Interface.wsdl` file in WebSphere Studio and use the editor features to update the files.
You can add port types, operations, messages, and message parts using the context menu for the next highest element. For example, to add a new operation, right-click **ProcessPortType** and select **Add Child → Operation**.

**Build the business process**
To work with the process, open the .bpel file. A default flow-based process built during the New Business Process wizard is shown in Figure 5-4 on page ccxxii.
Figure 5-4  New business process

The top box represents the process (zyxprocess in this case). Selecting this box will open the properties for the process in the window below.

Non-interruptible versus interruptible processes:

Non-interruptible processes are short running and stay active from the time they start to the time they complete. They manage state and status in memory. All activities in a non-interruptible process execute within a single transaction.

Interruptible processes can be suspended and resumed. Interruptible processes contain multiple transactions with each activity running in its own transaction. They persist their state and status to disk between activities. With this persistence, the process is forward-recoverable. When the process is suspended, execution can continue from the suspended state when the proper event or a time-out occurs. A process must be interruptible if it contains asynchronous activity or will execute for an extended period of time.
Add activities

Adding activities to the process is a simple matter of clicking the activity type in the palette and then clicking in the canvas area.

- When working with sequence-based processes, clicking the location for the new activity will automatically add links to the activities above and below.
- When working with flow-based processes, you add the links manually using the activities context menu. Links in a flow-based process have properties that allow you to add logic to determine the flow of the process.

With the new activity selected in the canvas, use the details area below to update the properties of the activity. You can use this window to name the activity, associate it with a partner link, add compensation, define staff activity, and any other settings relevant to the activity type.

Table 5-1 shows the primary set of activities available.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Invoke" /></td>
<td>The Invoke activity performs an operation. The operation is defined by a partner link and may be synchronous or asynchronous.</td>
</tr>
<tr>
<td><img src="image" alt="Receive" /></td>
<td>The Receive activity waits for an external input to the process before continuing. The operation supported by the receive activity is defined by a partner link.</td>
</tr>
<tr>
<td><img src="image" alt="Reply" /></td>
<td>The Reply activity sends a message to the partner defined by a partner link. This is typically used in processes that need to return a message to the partner that instigated the process.</td>
</tr>
<tr>
<td><img src="image" alt="Pick" /></td>
<td>The Pick activity waits for an incoming message and selects a path appropriate to the first message received. A time-based path can be configured to manage situations where no message is received. A partner link is associated with each message path.</td>
</tr>
<tr>
<td><img src="image" alt="Staff" /></td>
<td>The Staff activity delegates a task within the process to a human. The user interface in this case is either a custom application based on the process choreographer API, or the Web client provided that comes with WebSphere Business Integration Server Foundation.</td>
</tr>
<tr>
<td><img src="image" alt="Transform" /></td>
<td>The Transformer activity maps the contents of one or more message types to the contents of another.</td>
</tr>
<tr>
<td>Activity</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>![Assign]</td>
<td>The Assign activity copies information from one part of the process to another.</td>
</tr>
<tr>
<td>![Switch]</td>
<td>The Switch activity evaluates the conditions on a series of control paths and follows the first one that matches.</td>
</tr>
<tr>
<td>![While]</td>
<td>The While activity repeats the activities that it contains as long as a condition is met.</td>
</tr>
<tr>
<td>![Wait]</td>
<td>The Wait activity stops the process until a point in time has occurred or a time interval has elapsed.</td>
</tr>
<tr>
<td>![Sequence]</td>
<td>The Sequence activity defines a serial control path within a process.</td>
</tr>
<tr>
<td>![Flow]</td>
<td>The Flow activity defines a potentially parallel control path within a process.</td>
</tr>
<tr>
<td>![Terminate]</td>
<td>The Terminate activity stops the process immediately without performing any compensation or fault handling. The behavior of this activity depends on the location within the process.</td>
</tr>
<tr>
<td>![Throw]</td>
<td>The Throw activity signals that an error has occurred. This is typically handled by a fault handler element associated with a higher level of process structure.</td>
</tr>
<tr>
<td>![Empty]</td>
<td>The Empty activity element does nothing. It can be used as a placeholder during process design, and then changed to the appropriate activity when the process is implemented.</td>
</tr>
<tr>
<td>![JavaSnippet]</td>
<td>Java code can be embedded into the process using the Java snippet activity. While it is possible to embed business logic into this type of activity it is not advisable, as it removes the clarity of the process modelling. Snippets are designed to perform lightweight utility activities such as data mapping.</td>
</tr>
</tbody>
</table>

Activities also have context menus that allow you to add links between activities (flow-based), select the partner link for the activity, select the request and response variables, etc. The context menu will vary with the activity type.
Add variables

Variables represent messages in a WSDL file. Variables can be added by clicking the + symbol in the variables box. You also have the opportunity to define variables as you create activities.

To add a variable:

1. Click the + sign in the Variables area. This will add a new entry to the Variables table.
2. Name the variable by typing in the new name on the Variables area, or on the Description tab in the Details area.
3. Browse to the WSDL file containing the corresponding message and select the message in the drop-down.
Add partner links

The process will need partner links defined for the interface to the process and for the partners that provide services to the process.

A partner link can have one or more partner link types that define the interface between two services by specifying the messages and the port types that are used by each. Each partner link type is defined by specifying the WSDL file for the process or service being defined, the port type from the WSDL, and the role(s) for the partner link.

You have the option of creating one or two roles. Generally, you would use a single role for a request/response operation because the results are returned using the same operation, and the roles remain the same throughout the transaction. Conversely, you would use two roles for an operation that is asynchronous so that you can use the same partner in the callback operation by switching the role that it plays.
Chapter 5. WebSphere Business Integration Server Foundation architecture and overview

A default partner link for the process is included in the business process when you create it. You can modify this partner link or delete it and recreate it to reflect the needs of the process.

5.1.4 Deploying the business processes

Business processes can be deployed and tested to a remote WebSphere Business Integration Server Foundation application server with the BPE container installed, or it can be tested using the Integration Test Environment server in WebSphere Studio Application Developer Integration Edition.

The steps required to deploy the business process to the integrated server test environment in WebSphere Studio are:

1. Generate the deploy code (this creates an enterprise application (EAR) file for deployment).
2. Create a test server.
3. Add the EAR file for the business process to the server.
4. Publish and start the server.
5. Test the process with the Business Process Web Client.

The steps required to deploy the business process to a remote server are:

1. Generate the deploy code.
2. Export the EAR file to a location accessible to the server.
3. Use the WebSphere administrative console to install the EAR file.

Generate the deploy code

Preparing the business process for deployment from WebSphere Studio is made simple through the use of a wizard (Enterprise Services → Generate Deploy code). During the generation of the deploy code, the inbound bindings will be created and the partner links will be wired to the actual services that they represent. You can generate deploy code with one of three inbound bindings:

- EJB binding
- SOAP binding
- JMS binding

The results of the deploy code generation wizard will be a deployable enterprise application, including the necessary Web and EJB projects.
5.1.5 Accessing the process with the Business Process Web Client

WebSphere Studio provides a Business Process Web Client that provides access to a running process. Although it can be used as a general purpose user client, it is quite likely that a specialized application will be created for end users of the system. The client is useful during process development and testing to instantiate a process with a certain set of input parameters. For this discussion, we will refer to it as the Web client.

The Web client is installed when you install the BPE container, and runs as an application on the server. It can be accessed using the following URL:

http://<server_hostname>:9080/bpe

The Web client can be seen in Figure 5-7.

![Business process Web client](image)

Figure 5-7  Business process Web client

The actions you can take with the Web client depend on the role that you have been assigned; there are roles for processes and roles for work items.
The following is a brief outline of the tasks you can perform using the Web client:

- **Process templates**
  A process template is a deployed and installed process that is used as a template for process instances at run time. It contains all information of a process model.

  The Web client can be used to display, start, or stop business templates. If a process template is stopped, no new process instances of the corresponding process can be created.

  Process templates can also be started and stopped from the WebSphere administrative console. Process template options are found under Business Processes in the Related Items table for the application’s EJB module.

- **Process instances**
  A process instance is an instance of a process template. Multiple process instances of the same process template can be executed in parallel.

  You use the Web client to:
  - Display information about a process instance.
  - Work with process instances you administer.
  - Work with process instances you started.
  - Monitor a process instance.
  - Administer compensation for a process instance.
  - Terminate a process instance.
  - Query process instances.

  To debug or test a business process, you can use the Web client to start a process instance and to act as the client for the process instance, providing input to and receiving the output from the instance.

- **Work items**
  Work items are created when human interaction is required in a business process instance. If you are authorized as a potential owner, editor, or reader of an activity, the work item is added automatically to your To Do list, which is displayed in the My To Dos page of the Web client.

  When working with work items, you can do the following in the Web client:
  - Display work items in your To Do list.
  - Claim a work item.
  - Complete a work item.
  - Query work items.
  - Display information about an activity.
  - Create, transfer, or delete a work item.
The BPEL editor provides interfaces to customize certain pages for the Web client. The following is a list of activities with customizable pages:

- Staff activity
- Receive activity
- Reply activity
- Pick activity

The customized pages are associated with the activity in the Client tab of the properties for the activity.

### 5.1.6 Business Process Execution container (runtime)

The Business Process Execution (BPE) container is a specialized J2EE application that executes business processes and flows. It handles the lifecycle of a process, from instantiation of a process template to final deletion of the completed process.

It relies on a database to manage state and on a JMS provider for transitions between activities. This means that to make a fully scalable, available installation of the process choreographer, the characteristics of the underlying infrastructure must also be considered.

Figure 5-8 shows the components of the container. It is implemented as a J2EE application that uses the underlying WebSphere Application Server runtime services and resources.

![Business process execution container architecture](image)
The key components of the BPE container are:

- Process navigation
- Factory
- People interaction
- Internal interface
- External interface

**Process navigation**
The process navigation component consists of the navigator and some plug-in components.

**Navigator**
The navigator is the heart of the BPE container. It manages the state of all process instances and the activities that they contain.

The life of a process instance begins with a start request. This creates the instance based on a process template and puts it into a *running* state. When all its contained activities have reached an end state, the process instance is marked *finished*. The instance remains in this state until it is deleted, either implicitly or via an explicit API call.

The process instance might encounter a fault that was not processed as part of the process logic. In this case, it waits for the completion of the active activities before putting the process into its *failed* state. Compensation is then invoked if it was defined for the process.

A process instance can also be terminated by a process administrator. In this case, after completion of the active activities, the process instance is put into its *terminated* state.

**Plug-ins for process navigation**
The core capabilities of the navigator are extended using plug-ins. These provide future flexibility and extensibility for the product.

Plug-ins are provided for:

- Invocation of activity implementations
  There are currently two plug-ins that support invocation: The invocation of external processes via WSIF and the invocation of Java snippets.
- The handling of data in the process, such as evaluating conditions
  The process engine has a plug-in that understands conditions written in Java against WSDL messages.
The logging of events in an audit trail
The process engine has a plug-in that writes data to the audit trail table of the process engine’s database.

Factory
The factory component is responsible for state data that the process engine deals with. It allows data to be stored in one of the following forms:

- Transiently in memory
  This is used to support the efficient execution of non-interruptible processes
- Persistently in a database
  This is used to provide durability to interruptible processes. Many popular databases are supported, including DB2 Enterprise Server Edition.

People interaction

Note: Process choreographer supports business processes with people interaction only when WebSphere Application Server security is enabled. This is because the user needs to be authenticated to determine the appropriate work items.

The main components involved in interaction with people are:

- Web client or other client
  It is possible to interact with the process instances via the Business Process Web client. This client can be tailored to the requirements of the business application.
  Alternatively, the process choreographer API can be used to create a custom client.
- Work item manager
  The work item manager component is responsible for handling the work items created when the BPE container encounters a staff activity. This entails:
  - Creating and deleting work items
  - Resolving queries from process participants
  - Coordinating staff queries
  - Authorizing activity on process instances, ensuring that participants only gain access to process instances for which they have a valid work item

The work item manager has a number of performance-related features, notably an internal cache for resolved staff queries.
Staff support service, staff resolution plug-ins, and staff repositories

The staff support service manages staff resolution requests on behalf of the work item manager. It delegates execution to the staff resolution plug-ins that work with the staff repositories to fulfill requests.

Internal interface
Interruptible processes use a JMS queue between activities to provide durability. In most production environments this should be based on a robust external JMS provider, such as WebSphere MQ.

External interface
The interface to the container is via a façade. This is provided both asynchronously as a message-driven bean and synchronously as a session EJB.

5.2 Programing Model Extensions

WebSphere Business Integration Server Foundation provides a number of valuable extensions to the J2EE specification. These are delivered in many different forms, including services, APIs, and tooling extensions.

Business object model
These extensions target the business object model, that is, the representation of business data by a network of entity EJBs. These features provide extensions to the EJB 2.0 specifications and to the data access and caching functions supported by the base WebSphere Application Server. They include:

- Application profiling and access intent
- Dynamic query service
- Container-Managed Persistence over Anything

Business process model
For the business process model:

- Business Process Choreographer using BPEL4WS
- Last participant support
- Activity session service and extended EJB lifecycle
- Business Rule Beans

Next generation apps
For next generation apps:

- Extended Messaging Support (EMS)
- Asynchronous beans
5.2.1 Container-Managed Persistence over Anything

The container-managed persistence (CMP) features of WebSphere Application Server include those defined by the EJB 2.0 Specification, as well as capabilities that are beyond the specification.

As specified by the EJB 2.0 Specification, the WebSphere Application Server EJB container automatically manages the state of CMP entity beans. This management includes synchronizing the state of the bean with the underlying database when necessary, and also managing any relationships (CMRs) with other entity beans. The bean developer is relieved of writing any database-specific code and, instead, can focus on business logic.

Several capabilities are provided to enhance the function of CMP entity beans that go beyond those capabilities defined by the specification. These include:

- Entity bean inheritance, enabling a developer to define fields, relationships, and business logic in a superclass entity bean that are inherited by all subclasses
- Access intent policies, which provide a mechanism to indicate the intent of an application's interaction with the essential state for entity beans in order that the persistence mechanisms can make appropriate optimizations
- Caching data across transactions to improve performance (LifetimeInCache)

WebSphere Application Server has long provided CMP support for persistence to relational back-end systems. Now, with WebSphere Business Integration Server Foundation's CMP over anything (CMP/A) feature, this support has been extended to the following non-relational back-ends.

- JCA adapters (CCI)
- Web services (JAX-RPC)
- Services-oriented architecture (WSIF)
- EJB (RMI-IIOP)
- Stored procedure (JDBC and SQLJ)

The use of CMP/A has the following advantages over the traditional relational database support or bean managed persistence:

- It expands the back-end systems that can be used for container managed entity EJB persistence. Although bean managed persistence was available for
using non-relational back-ends, it required manual coding. CMP/A support now does most of the work for you.

- CMP optimization tooling is provided to facilitate the use of application profiles and access intents. These facilities have been available for entity EJBs, but the implementation was not intuitive. The optimization tooling creates a set of application profiles based on a scan of the code that can be used as is or modified.

The CMP infrastructure is designed to work with the J2EE Connector Architecture (JCA) through an interface called EJBTtoRAAdapter. In addition to the traditional implementation of this interface (which connects to relational back-end systems), CMP/A provides support for connecting to procedural back-end systems. The new procedural implementation contains generic logic for adapting between JCA and the WebSphere Application Server persistence manager.

WebSphere Studio Application Developer Integration Edition and the EJB Deployment tool have been updated so they can be used to return either the traditional CMP implementation or the new CMP/A implementation. The back-ends are defined at the bean level, but different beans in the same JAR may use different back-ends.

Figure 5-9 on page ccxxxvi shows a runtime view of a CMP/A implementation.
The development steps are:

1. Create a CMP entity EJB using the wizards in WebSphere Studio Application Developer Integration Edition. This is the same process as previously used to create any entity EJB.

2. Define any business logic push-down methods as abstract in the bean class, and promote those methods to the interfaces. Note that defining a business logic method as abstract violates J2EE specifications, making CMP/A support an extension to the specifications.

3. Update the EJB deployment descriptor to map the business logic push-down methods to a corresponding function provided by the back-end.

The information required depends on the back-end type selected. For example, if you select JDBC as a back-end type, you will need to select which push-down method to use and enter the JDBC call. If you select JAX-RPC as a back-end type, you will need to define the WSDL file name and then map an operation in the WSDL file to the push-down methods.

WebSphere Studio Application Developer Integration Edition provides a Pushdown tab for this purpose in the Deployment Descriptor editor.
4. Generate the push-down implementation (using the Generate button on the Pushdown tab). This will generate skeleton Java files, `<beanName>PushDownMethodsImpl` and `<beanName>PushDownMethods`, containing methods corresponding to the business logic push-down methods and the `<beanName>PushDownMethods` Java file.

5. Update the code in `<beanName>PushDownMethods` to complete the implementation. Comments in the skeleton will guide you as to the required modifications.

6. Generate the code for deployment using EJBDeploy. The EJBDeploy tool will recognize the CMP/A definitions in the deployment descriptor extensions and will generate the correct set of deployment classes for the non-relational persistence.

5.2.2 Application profiling and access intent

Entity EJBs are reusable components by nature, but this reusability often introduces a performance problem. It is common to have different clients access the same set of entity EJBs, yet the intent of each client is completely different. With WebSphere Business Integration Server Foundation, the EJB container can be configured to provide optimal performance for entity bean access based on a specific type of use. This is done by declaring access intent hints at assembly time that guide WebSphere resources to provide the appropriate access intent services for every EJB request. These hints, organized as access intent policies, are used by the following components as the optimize entity bean access:

- The Persistence manager, as it makes decisions about isolation level, cursor management, and so on.
- The EJB container, as it influences the management of EJB collections.
- Relational Resource Adapters (RRAs), as they provide prefetch hints in defined increments to control the number of rows read from a database at a time. WebSphere does not provide paging nor does the RRA provide prefetch. Instead, a prefetch hint is passed to the database telling how many rows are going to be read so that the database can optimize the access.

**Access intent policy**

An access intent policy is a named set of properties (access intents) that govern data access for EJB persistence. You can set access intents only within EJB Version 2.x-compliant modules for entity beans with bean-managed persistence or with CMP Version 2.x.

Access intent policies are defined by the following attributes:

- Access type, defining the type of access (read, update, etc.), and concurrency control (locking).
- Collection increment and scope that determine how a collection returned from a finder method is loaded and retained in memory.

- Resource manager prefetch increment that suggests how many rows should be retrieved from the underlying relational database in a single operation. It is a hint to the JDBC driver and may or may not be taken into account.

- Read ahead, used when there are multiple entity EJBs with a Container Managed Relationship (CMR) defined on them. Possible values are the names of any CMR field on the current entity EJB. The container uses the read-ahead parameter to prefetch the data of the related entities. This only takes effect when the loading of the entities takes place as a result of invoking a findByPrimaryKey() method.

Access intent policies can be applied at the bean or method level. At the bean level, they represent the default access intent policy for methods. Access intents defined at the method level can be used to assign different access intent policies to an EJB’s create, remove, setter and getter methods.

WebSphere Business Integration Server Foundation comes with several predefined access intent policies.

**Application profiling**

Application profiling lets you associate task names to session and entity EJB methods, and group a set of tasks under a set of access intents. This provides the capability of associating an entity EJB method with a specific access intent policy at runtime, depending on the task under which it is called. So the binding is caller dependent and resolved dynamically at runtime.
A task is associated with an application profile at deploy time. At runtime, the task name gets propagated, and its arrival identifies the caller. An application profile associated with that caller task defines the access intents to be applied.

**Profiling prioritization**

When different access intents are defined on the same entities, prioritization is applied to resolve which access intent to apply. Figure 5-11 on page ccxl shows the decision algorithm used to apply an access intent when many access intents have been defined or are applicable for the current request.
Figure 5-11  Application profiling prioritization
Steps to using application profiling and access intents

Application profiling requires accurate knowledge of an application's transactional configuration and the interaction of the application with its persistent state during the course of each transaction. Application profiling and access intent are meant to be configured at deployment time through the extended deployment descriptors. For flexibility, an API is exposed for entity BMP EJBs.

WebSphere Studio Application Developer Integration Edition and the Assembly Toolkit both provide tools for configuration of application profiles and access intent policies.

The steps to be taken to prepare an application for profiling are:

1. Create a custom access intent policy to fit your needs
2. Create a container task.
   Associate the task with EJB methods.
3. Create an application profile.
   a. Associate the profile with a set of tasks.
   b. Create a set of access intents.
      Associate entity EJB methods with a custom or predefined access intent policy.

Automatic configuration: WebSphere Studio Application Developer Integration Edition and the Assembly Toolkit provide a tool for automatic application profile configuration. The tool will analyze the application and build an application profile configuration for you. This tool has its limitations, but can provide a starting point for building the appropriate application profiles.

5.2.3 Dynamic query service

The EJB query language (EJB QL) is used to specify queries to container-managed entity beans. The query language, based on SQL, allows searches on the persistent attributes of an enterprise bean and associated bean attributes. An EJB query is independent of the bean's mapping to a persistent store. WebSphere Application Server supports the standard EJB QL specifications, with some extensions that increase the flexibility of the syntax of EJB QL.

EJB QL allows querying of container-managed persistence (CMP) and container-managed relationship (CMR) fields. An EJB query can be used to
define a finder or select method of a CMP entity bean. Finder and select queries are specified in the bean's deployment descriptor using the <ejb-ql> tag.

Queries specified in the deployment descriptor are compiled into SQL during deploy, making EJB QL static by nature. Once an EJB QL query is associated with a finder method and the EJBs deployed, the query can only be modified by redeploying and re-installing the EJB.

The dynamic query service removes this limitation. It allows applications to formulate queries at runtime and have them submitted and executed on the fly, providing a behavior that resembles that of traditional dynamic SQL. Dynamic query service maintains the same features of EJB 2.0 QL and of the base WebSphere query enhancements, while providing even more functionality.

Table 5-2 compares the query support provided in the EJB 2.0 specification, WebSphere Application Server, and WebSphere Business Integration Server Foundation dynamic query service.

Table 5-2 Query support in WebSphere

<table>
<thead>
<tr>
<th>Query extensions</th>
<th>EJB 2.0 Query</th>
<th>Base WebSphere Query</th>
<th>Dynamic Query Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select clause</td>
<td>Required</td>
<td>Optional</td>
<td>Optional</td>
</tr>
<tr>
<td>Delimited identifiers</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>String comparisons *</td>
<td>= and &lt;&gt;</td>
<td>= &lt;&gt; &gt; &lt;</td>
<td>= &lt;&gt; &gt; &lt;</td>
</tr>
<tr>
<td>Scalar functions</td>
<td>Some</td>
<td>More</td>
<td>More</td>
</tr>
<tr>
<td>Calendar comparisons</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Order by *</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SQL date/time expressions</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Query over inheritance graph</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Subqueries</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EXISTS predicate</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DISTINCT predicate *</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Aggregate functions (MIN, MAX, SUM, etc.) *</td>
<td>No</td>
<td>In subquery</td>
<td>Yes</td>
</tr>
<tr>
<td>GROUP BY, HAVING clauses</td>
<td>No</td>
<td>In subquery</td>
<td>Yes</td>
</tr>
<tr>
<td>Bean business methods in query</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
* These functions are included, to some extent, in the EJB 2.1 specifications.

The use of the dynamic query service in an enterprise application has the following benefits:

- There is no need to know the query search criteria until application runtime.
- Dynamic query service allows you to return multiple CMP or CMR fields from a query, while deployment queries allow only a single element to be specified in the SELECT clause.
- Dynamic query service allows you to perform aggregation in the query, while deployment queries do not allow use of the aggregation function SUM, AVG, COUNT, MAX, MIN in the top level SELECT of a query.
- Dynamic query service allows you to use value object methods or bean methods in the query statement.
- In some cases, using the dynamic query service may alleviate performance issues with EJB QL queries. Using the dynamic query service, you can formulate queries that would require a lot of hand-coded processing if you used the standard EJB QL. For example, assume that you have a query that returns the maximum salary of employees, on a department-by-department basis. With dynamic query service, you can formulate and execute your query, and get exactly the result you need, without further processing and without requiring EJB activation. With regular EJB QL, your program would have to manually do the grouping and aggregation, and EJBs would have to be massively activated.
- Dynamic query service may reduce the need for specific finders. You may add finding behavior to an existing application without even redeploying it.
- You can use the dynamic query service when you want to interactively test an EJB query during development but do not want to repeatedly deploy your application each time you update a finder or select query.

The dynamic query service is implemented using *push-down* technology. This technology involves taking an object query statement and using metadata that describes the mappings of EJB attributes to database tables and columns, to translate it to an SQL statement that can be executed by the database.
management system. In effect, the EJB query is pushed down to the database, which is designed to perform this type of query evaluation, and groups and sorts very efficiently. As would be expected, not all of an EJB query can be pushed down. The application server must still evaluate some query criteria, such as results of method invocations. By delegating most of the work, including navigation of relationships, to the database, excellent performance can be provided. Lastly, this push-down technology is designed to be independent of the database vendor. It works with any relational database supported by WebSphere.

**Dynamic query bean API**

The dynamic query service is provided as a special stateless session bean, named QueryBean. The client of the query session bean may be a remote client or it may be a local client, depending on whether the client makes use of the bean’s remote or local interfaces.

A remote client accesses the query bean through the bean’s remote interface, and remote home interface. A remote client can be any Java program such as an application, JSP, applet, or servlet.

A local client accesses the query bean through the bean’s local interface and local home interface. A local client is collocated in the same JVM with the query bean and can be another enterprise bean such as a session bean, entity bean, or message-driven bean.

The Query Engine can perform queries on entity beans, on CMP fields, and on CMR fields.

The QueryBean interface has three client methods:

- `executeQuery()`: This method parses and executes the query in a single operation. The `executeQuery` method in the remote interface has some extra arguments over the method provided by the local interface.

- `prepareQuery()`: This method invokes the query parsing and plan creation process. It receives exactly the same three common input parameters described above (queryStatement, parameterVars, and queryDomain). This method returns the optimized query plan generated from the query string input parameter. Normally a client would not directly invoke the `prepareQuery` or `executePlan` methods. The return type of this method is a string containing the created query plan, which is a query statement parsed, validated, and optimized, and it is presented as an input to the `executePlan` method.
executePlan(): This method executes a query plan that is in a string text form. It receives two input parameters for the local interface, and an additional two for the remote interface.

**Runtime and development support**

The QueryBean is packed in query.ear file that is automatically installed on the default server during the WebSphere Business Integration Server Foundation installation. If new application servers are created, the query.ear file must be manually installed on each additional application server in the same fashion as any other application is installed.

For development purposes, query.ear must be imported into the project where QueryBean is used. Client interfaces to the QueryBean must also be imported.

Table 5-3 shows the JAR files that comprise the dynamic query service.

<table>
<thead>
<tr>
<th>JAR File</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>query.jar</td>
<td>Query parser, runtime, query session bean</td>
</tr>
<tr>
<td>qryclient.jar</td>
<td>Client stubs and classes</td>
</tr>
<tr>
<td>query.ear</td>
<td>Query session bean application (located in installableApps, it should be installed in application server runtime environment)</td>
</tr>
</tbody>
</table>

**5.2.4 Last participant support**

Last participant support extends J2EE transaction support to allow a transaction involving several two-phase commit resources and a single one-phase commit resource. This makes it possible for a resource adapter that does not implement the XAResource interface to participate in a global transaction. At transaction commit, the two-phase commit resources will first be prepared. If this is successful, the one-phase commit resource will be called to commit, followed by the call to commit for two-phase commit resources.

Multiple interactions may occur involving the one-phase commit resource in the same transaction, but only one such resource may be involved. See Figure 5-12.
Figure 5-12  Last participant support

Last participant support introduces the hazard of a mixed resolution of the global transaction if the one-phase commit resource succeeds and one or more of the two-phase commit resources fail during the commit phase. The one-phase commit resource cannot be recovered.

Last participant support cannot be considered as a substitute for the two-phase commit protocol. Applications that use this feature must be structured to handle the hazard mentioned before.

**Heuristic reporting**

Using last participant support introduces the hazard of a mixed output if one or more two-phase resources fail to commit in the commit phase. At this moment, the prepare commit has been issued, and the one-phase commit resource has been committed. No rollback can be performed on the one-phase commit resource. The output is mixed, as there will be committed and uncommitted resources. A more unusual situation is the heuristic condition that happens when the system crashes when resolving the one-phase commit resource. In this case WebSphere Application Server will be unable to determine how the one-phase commit resource was resolved. The two-phase commit resources will be rolled back.
Heuristic reporting provides tracking of one-phase commit resources so that you can be sure if these resources have been correctly resolved or a mixed output state has been reached. With heuristic reporting enabled, a line is written to the log before and after committing the one-phase resource. If any of the XA resources cannot commit, it will provoke a rollback in all the two-phase commit resources. The heuristic reporting will inform you whether the one-phase commit resource succeeded in committing, provoking a mixed outcome, or was also rolled back.

**Enabling last participant support**

Last participant support is enabled in the application deployment descriptor extensions. In WebSphere Studio Application Developer Integration Edition or the Assembly Toolkit, the checkbox to enable it can be found on the Extended Services tab.

Heuristic logging is enabled at the application server level in the transaction service properties.

### 5.2.5 Activity session service and extended EJB lifecycle

ActivitySession support extends J2EE transaction support to allow a transaction involving several one-phase commit resources.

A Resource Manager Local Transaction (RMLT) is a resource adapter's view of a local transaction. It represents the unit of recovery on a single connection that is managed by the resource manager. With the standard J2EE support, EJBs cannot extend an RMLT beyond the boundary of an EJB method invocation. A J2EE application accessing one or more EJBs backed by non-transactional one-phase commit resources is not capable of coordinating these resources.

A Local Transaction Containment (LTC) is a bounded unit of work scope where zero, one, or more RMLTs may be accessed. The LTC defines the boundary at which all RMLTs must be complete. Any incomplete RMLT may be resolved according to the resolution policy specified at deployment time. An LTC context is always established by the container in the absence of a global transaction. An LTC is local to a bean instance. LTCs are not shared across beans even if those beans are managed by the same container. The J2EE specification defines the LTC on the bean method level.

ActivitySession provides both a way to extend transaction boundaries beyond method invocation and to specify EJB activation and passivation by means of these new boundaries. ActivitySession becomes a new entity in the transactional scenario defining new boundaries for a unit of work. ActivitySession can be associated with an HTTP session, so not only can the unit of work boundaries be defined by the client, but the EJB life cycle can be scoped to that client.
Figure 5-13 shows how ActivitySession can extend the LTC. In this particular scenario, the ActivitySession is scoped to the client life cycle. Entity EJBs could be activated at the beginning of the ActivitySession and passivated at the end, reducing the overhead of activating and passivating the EJBs on each method call.

There are a few usage rules that have to do with RMLTs, LTCs, global transactions, and ActivitySessions:

- Nested ActivitySessions are not supported.
- ActivitySession boundaries cannot overlap.
- ActivitySessions may encapsulate one or more global transactions.
- Global transactions within an ActivitySession are independent of each other.
- It is not possible for an LTC to coexist with a global transaction.
- No one can wrap an ActivitySession; ActivitySessions exist on their own.

**Extended local transaction**

ActivitySession service extends the J2EE LTC beyond the method invocation, providing extended local transactions. Extended local transactions are offered in WebSphere Business Integration Server Foundation by means of the ActivitySession as LTCs.
There are two patterns of LTC usage:

- **Containment**
  
  RMLTs within the LTC are started and completed by the application. RMLTs are said to be *contained* by the LTC. RMLTs that are not completed by the application by the end of the LTC boundary are cleaned up by the container in a direction (commit or rollback) determined by the unresolved action policy. This is a bean-managed or programmatic approach.

- **Enlistment**
  
  RMLTs are started by the container when the application first uses a connection. RMLTs are said to be *enlisted* in the LTC. RMLTs are completed by the container at the end of the LTC. This is a container-managed or declarative approach.

The ActivitySession service captures the commit operations of each local and global transaction and upholds them. At the end of the ActivitySession, the container will effectively commit or roll back each individual transaction.

The ActivitySession is not a substitute for the two-phase commit protocol. An ActivitySession may result in a mixed outcome, if any single-phase resource successfully gets committed before another resource fails to commit. In this case, the ActivitySession service will allow the programmer to retrieve the list of resources that were committed and those whose state is uncertain.

Components that support ActivitySession are:

- EJB components: Session and entity beans
- Web components: Servlets

**Configuration**

Configuration consists of defining how the activity session is going to behave in each component and what level of support is expected by each method. WebSphere Studio Application Developer Integration Edition exposes these settings in the deployment descriptor extensions with the Deployment Descriptor editor.

For EJB components, you can configure the following:

- Resolution control settings (Beans tab) determine whether the container takes care of starting and committing local transactions (enlistment), or the application does (containment).

- Unresolved action settings (Beans tab) define whether unfinished transactions will be committed or rolled back when an activity session is complete.
Boundary settings (Beans tab) identify the containment boundary at which all contained RMLTs must be completed (bean or activity session).

The level of support (Activity Sessions tab) expected by each method (never, supports, requires new, etc.).

Entity beans are always container-managed from an activity session standpoint, eliminating the need to specify activity session attributes. Entity EJBs will use any activity session context provided by their callers.

Session beans will support an activity session type that matches the transaction type. If the session bean supports container-managed transactions, then it supports container-managed activity sessions. If the transaction type is bean-managed, then the activity session needs to be started by the bean using the ActivitySession API.

For Web components, you can define whether the activity session control lies with the application (using the ActivitySession service API) or with the container. If you choose the container, the container will automatically start and attach a new activity session every time a new HTTP session is created.

**ActivitySession service API**

You can achieve the same goals using ActivitySession programmatically using the API instead of configuring container ActivitySessions. The ActivitySession service is exposed to application programmers through the com.ibm.websphere.ActivitySession.UserActivitySession interface.

- An ActivitySession is started with a beginSession method call and ended with a endSession method call.
- The UserActivitySession lets you checkpoint work to commit changes made up to a specific point without ending the ActivitySession.
- The resetSession method call causes the ActivitySession service to roll back all changes to the last checkpoint.
- You can programmatically set a timeout for the ActivitySession through the setTimeOut method call.

The implementation of this interface can only be accessed through a JNDI lookup. The UserActivitySession implementation is bound to the java:comp/websphere/UserActivitySession name.
Extended EJB life cycle
Without ActivitySession service support, there were only two different ways to keep an EJB within scope between multiple method calls:

- Use of commit option A.
  This option prevents the EJB from being workload managed and disallows sharing the database tables with other applications.
- Start a global transaction.
  Global transactions need external resources (not only from the resource manager) to coordinate the different RMLTs. A transaction service such as the JTS must be used. This carries a performance weight.

ActivitySession service lets you extend the lifecycle of an EJB without the performance drawback of a global transaction and avoiding the restrictions of using commit option A.

An EJB activation can be scoped to the ActivitySession, rather than being scoped to the transaction boundaries, offering additional control on the timing for EJB activation and passivation, with potential performance benefits.

This feature can highly improve performance by avoiding unnecessary passivations. EJBs remain active and ready.

New EJB Bean cache policies have been added to existing IBM extensions. The Bean cache policies are governed by two attributes:

- Activate At: Defines when an EJB is activated and placed in cache, as well as when it is removed from cache and passivated (once, transaction, ActivitySession).
- Load At: Specifies when the container should synchronize the bean with the persistence layer (activation or transaction). This implies whether the container has exclusive or shared access to the database.

The Bean cache policy governs which commit option will be used for the specific bean where it is configured.

<table>
<thead>
<tr>
<th>Load At</th>
<th>Activate At</th>
<th>Once</th>
<th>Transaction</th>
<th>Activity Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation</td>
<td>A</td>
<td>Unsupported</td>
<td>Unsupported</td>
<td></td>
</tr>
<tr>
<td>Transaction</td>
<td>B</td>
<td>C</td>
<td>C+</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-14 Bean cache policies and commit options
Figure 5-14 shows which commit options will be used depending on the Load At and Activate At values of the bean cache policy.

According to the EJB 2.1 specification, supported commit options are A, B and C. Commit option C+ is an IBM extension to support an EJB life cycle bounded to the ActivitySession boundaries. With commit option C+, the EJB is activated only once per ActivitySession, but data may be loaded or stored multiple times within that session at checkpoints.

- **Option A**
  
  When the transaction ends, the container caches the instance and it remains valid across transactions. The container invokes ejbStore to synchronize the instance. The instance remains both ready and valid, so the container will not load it again from the persistent storage when the next transaction begins. Exclusive access to the instance is ensured by the container.

- **Option B**
  
  The container invokes ejbStore, caches the instance, but marks it as not valid. When a new transaction begins, the container invokes ejbLoad to synchronize the bean, since the container does not ensure exclusive access.

- **Option C**
  
  The container does not cache the instance. At the end of the transaction, the container invokes ejbStore and ejbPassivate and returns the instance to the pool. At the beginning of the next transaction, ejbActivate and ejbLoad are invoked on the retrieved instance to synchronize the state of the EJB.

- **Option C+**
  
  This option is different from the others because instead of referring to transaction boundaries, it refers to ActivitySession boundaries. An ActivitySession may contain multiple transactions, global or local. With commit option C+, an EJB is stored at the transaction end and loaded at the beginning of the next transaction. No activation or passivation is needed. The same object is reused between transactions.

These options are exposed in the Beans tab of the EJB deployment descriptor when editing it using the Deployment Descriptor editor in WebSphere Business Integration Server Foundation.

**Usage scenarios**

Using the ActivitySession service in association with a Web container lets you extend and define transactions of EJBs during the client’s life cycle.

Coordination of one-phase commit resources is not supported by the J2EE specification. ActivitySession adds more flexibility to the constrained J2EE
model, enabling the coordination of one-phase commit resources under a unit of work.

- **Client-scoped life cycle**
  
  ActivitySessions used within a Web container are automatically associated to the HTTP Session. An EJB can be activated and passivated in ActivitySession boundaries. Therefore, it is possible to span the life of an EJB and map it to the client’s life cycle.

- **Client-side demarcation**
  
  The association between the ActivitySession and the HTTP Session also reward you with the possibility of client-side defined transactions. ActivitySession lets you coordinate a number of one-phase commit resources within the boundaries of a single unit of work. You can span this unit of work through multiple client invocations.

- **Coordination of one-phase commit resources**
  
  ActivitySession service allows the coordination of multiple one-phase commit resources within the boundaries of a single unit of work. It is not a substitute for the two-commit phase protocol and introduces a heuristic hazard.

### 5.2.6 Business Rule Beans

The Business Rule Beans framework extends the scope of WebSphere Business Integration Server Foundation to support business applications that externalize their business rules. By externalizing rule processing, the rule can change without affecting the business process.

A business rule defines or constrains some aspect of a business by asserting control over some behavior of that business. A business rule officiates over frequently changing business practices, and can come from government regulations, company practices, customer status, or other external factors such as adapting business processes to react to competitive pressure. These types of rules can change periodically, and do not require a rebuild of all applications that support the business process.

At its simplest level, a business rule is little more than a well-placed if/then statement that compares a variable against a determined value, and then issues a command when they match. An application can access these business rules through Business Rule Beans. To access the Business Rule Bean, a program simply needs to implement a trigger point; this trigger point interfaces with the Business Rule Bean framework to execute the business rule that is encapsulated within a Business Rule Bean.
The Business Rule Beans framework not only decouples the business code from the application code, but also decouples user roles within an organization. In any organization applying the Business Rule Beans framework, the following three roles need to exist:

- **Rule-based Application Component Developer**: Provides EJBs and servlets that call out to business rules when appropriate. The component developer is conceptually unaware of the specific implementation that is going to be associated with the rule.

- **Rule Implementation Developer**: Creates the Java classes that provide the concrete implementation of the business rules.

- **Domain Expert (Business Analyst)**: Determines the values for the business rules, such as the start and end dates for using the rule, the rule’s initialization parameters, and the business intent. The Rule Management Application tool provides a simple administrative GUI that can be used by business analysts who are not IT professionals to configure existing rules.

This kind of working arrangement is advantageous because these individuals are able to work in parallel. The rule implementation developer may create a rule without needing to know the values that are to be contained inside it, and the analyst can modify it knowing the rule name and the folder it is located in, using the Rule Management Application.

Taking a closer look at the Business Rule Beans framework components, we have:

- **Business Rule Beans EJBs**, which provide a runtime environment for the framework
  
  During the development phase, they persist business rule configurations. At runtime they are responsible for finding and firing Business Rule Beans rules. Business Rule Beans EJBs are deployed with the application that uses them.

- **Business Rule Beans Trigger Point framework**, which provides an interface to access Business Rule Beans rules from the rule client (application code that uses a Business Rule Beans rule)
  
  Trigger points are pieces of application code that make a call to a rule. They are placed in EJBs, servlets, or Java classes that depend on business logic implemented in Business Rule Beans rules. The rule client creates an instance of a TriggerPoint object and calls one of its trigger methods, which looks up Business Rule Beans rules by names defined in the Rule Management Application. Once the rule is found, the TriggerPoint object invokes the rule by calling the fire method on the RuleImplementor (a Java class that implements a business rule).

- **Business Rule Beans Rule Implementors**, which are Java classes that provide implementation for common business logic
Programming a unique new rule implementation in Java is usually a simple process, made easier by the set of predefined rule implementors that can be used to create a customized business rule. These rule implementors are ready for use by rule clients through the TriggerPoint objects described above.

- A starter set library of generic rules, for example, GreaterThan, IsNull, and Range written in Java, that can be used for building customized rules
- An interactive Rule Management Application to create, retrieve, update, and delete rules
  It is a stand-alone tool for maintaining the rules. You can use it to configure the rule name, implementation class, start and end date, initialization parameters, etc.
- A batch eXtended Markup Language (XML) based rule loader/unloader to make rules portable
  This is the Business Rule Beans Rule Importer and Exporter tool, which allows you to import and export Business Rule Beans rule configuration as XML files. It can be used to move Business Rule Beans rule configuration between servers.

At runtime, the rule client invokes Business Rule Beans rules using the TriggerPoint object. Each Business Rule Beans rule configured in the Rule Management Application is represented as an EJB at runtime. After the Business Rule Beans rule is located, the framework fires the actual implementation and returns results to the rule client. Figure 5-15 on page cclv shows the interaction of Business Rule Beans components.

The runtime code that is used to find and trigger rules is made up of two parts:

- The rule client component. It can be any EJB, servlet, or Java class in your enterprise application that is dependent on the rule business logic. You can place the TriggerPoint in any of these components to use your Business Rule Beans rule. The client code is used to do the following:
  - Find a specified rule.
– Determine where the rule should be triggered.
– Call the fire method on all the rules
– Combine the results from the rules.

▶ The part that runs on the server consists of the EJBs used to represent rules and rule folders. These EJBs do the following:
  – Provide for business rule persistence.
  – Provide query functions that the client part of the runtime can use to find rules to be triggered.

The objects used to implement a business rule contain methods and attributes used by Business Rule Beans runtime and/or its administrative component. An externalized business rule is implemented as a pair of objects:

▶ Rule: This is an entity EJB that stores all the persistent data for the business rule. This is the object that the trigger point framework code performs a query to find the rule object(s) representing the business rules to be triggered. Once the rules are found, the framework code determines where the rule is to be invoked, either local to the trigger point or remotely on the application server. It then invokes the fire method on either the Rule EJB itself or on the local copy of the EJB to perform the function of the business rule.

▶ RuleImplementor: The class name of the business rule’s RuleImplementor is stored persistently in the rule. The RuleImplementor is a transient object (not managed by the application server) that the rule instantiates and then uses to do the actual work. When the fire() method is called on the rule object, the Rule object combines its persistent set of values with the parameters it received on invocation to create the parameter list for the parameter list. The actual execution of the RuleImplementor algorithm can take place either remotely or locally.

The Business Rule Beans framework uses a database to persist Business Rule Beans rule data. You can create a separate database to maintain rule data or create tables in the main database used by the application. All EJBs accessed in the same transaction must specify the same isolation level. Using a different database for business rules removes this restriction, because application EJBs and rule EJBs are accessed in different transactions. Having separate databases also follows the concept of externalizing the business rules from the enterprise application. Moreover, having the rules data separate from the application data facilitates the reuse of the existing business rules with other applications.

There are two kinds of rules in WebSphere. The difference between the two is mainly in the return value.

▶ A classifier is capable of accepting a set of input values, and returning a set of values that enumerates what something with those characteristics is. These rules are very useful for such things as controlling process flows,
implementing systems that are event driven, and other more advanced system architectures. A classifier rule returns a string that represents the classification. Typically, the values that the string can assume belong to a limited set (Gold, Silver, Platinum for customer status, for instance).

- A non-classifier or base rule is the most generic kind of rule. It processes the input parameters and returns an array of objects.

### 5.2.7 Extended Messaging Support (EMS)

Asynchronous messaging patterns are an important part of the J2EE application’s programming model because of the flexibility that they bring to a distributed application architecture. Asynchronous messaging systems allow for “loose coupling” between different applications, and between the different programs that constitute a single, distributed application. Importantly, these systems can span a wide number of platforms. Moreover, when sending and receiving messages, the senders and recipients of these messages do not need to know about each other directly. This lets them evolve independently of each other. These patterns are useful for process flows, parallel processing, time-independent processing, and event-driven processing.

Extended Messaging (EM) combines a number of development and runtime enhancements to make the delivery of messaging programs faster and more efficient in a WebSphere environment.

**Java Message Service**

The Java Message Service (JMS) is a set of Java APIs that provides a framework for Java programs to make portable calls into asynchronous messaging systems. However, JMS is a standard Java API, and does not take full advantage of the J2EE container. In addition, JMS is a relatively low-level API and requires the developer to write code for:

- Object lookups: JNDI lookups for connection factories and destinations.
- Message formats: The developer is tasked with constructing the parts of a message (such as XML), and then parsing these message parts upon retrieval.
- Caching of administration objects: For better performance, all thread-safe JMS objects should be cached.

The EJB 2.0 specification introduced message-driven beans (MDB), which provide container support for event-driven message retrieval. But the MDB specification still does not handle a number of messaging patterns, such as:

- Sending messages with or without a response.
- Application-callable receiver beans. Messages need to be retrieved during a program's application flow, as opposed to the MDB style, where messages are processed as soon as they arrive at a destination.

- Late responses in send and receive. This allows the sender to register an MDB type listener to handle responses that cannot be processed within a programmatically set timeout.

- Mapping and reformatting data in JMS messages. This lets the programmer set the details of the JMSMessage as parameters of a message, instead of having to build and parse the message data.

- Sending messages to multiple destinations (topics or queues).

**Extended Messaging**

Extended Messaging (EM) enhances standard J2EE Messaging by providing support for all types of messaging patterns, container support for these patterns, and code simplification.

One of the problems with JMS coding is the amount of code and J2EE constructs you have to write just to send a message. There is no container support for caching of connection factories, queues, senders, and so on. All that has to be done by hand. Extended Messaging promotes code simplification by:

- Providing advanced tooling support. All messaging patterns discussed can be supported by running wizards in WebSphere Business Integration Server Foundation.

- Data mapping of message parts.

- Sending messages to multiple destinations via Deployment Descriptor settings on input and output ports.

- Tooling support provides response queue handling.

- Runtime container caching of administration objects.

**Messaging patterns**

Extended Messaging is provided as a set of application interfaces that provide an abstraction for JMS. The following messaging patterns are supported by EM:

- Sending
  - Send - Fire and forget
  - Send with a synchronous response (optional timeout)
  - Send with a deferred response

- Receiving
  - Event style (MDB) with no response
  - Event style (MDB) with a response
– Application callable (optional timeout) with no response
– Application callable (optional timeout) with a response

**Sending pattern**

The sending pattern, illustrated in Figure 5-16 on page cclix, follows this scenario:

1. Client application looks up a sender bean and calls one of its sender methods.
2. The sender method formats the message parts into a single message.
3. The sender bean sends the message to an output port.
4. The WebSphere runtime maps the output port to one or more JMS destinations.
5. The WebSphere runtime sends the message to the destination defined for the output port.

![Figure 5-16 Sending process](image)

**Receiving pattern**

The receiving pattern has two forms:

- **MDB style**
  - The MDB listens to a JMS destination, receives the message, and then calls a worker session bean.
  - There is no input port for the MDB style.

- **Application callable receiver bean**
  - Receives the message by listening to the destination specified in the input port.
- Parses the message.
- Returns the message contents to the calling application.

Both receiving patterns are illustrated in Figure 5-17 on page cclx.

Figure 5-17  Receiving process

Runtime and development support
EM runtime support is delivered in WebSphere Business Integration Server Foundation. Because EM EJB beans run within a WebSphere Application Server runtime, they take full advantage of J2EE caching, clustering, transactional, and security constructs.

Extended Messaging programs are generally built using wizards in WebSphere Studio Application Developer Integration Edition, although the runtime interfaces are provided by the WebSphere Business Integration Server Foundation runtime and can be programmed by hand. Design considerations are similar to standard JMS and MDB applications except that EM will greatly simplify the coding of these applications. The output of the WebSphere Studio wizards will be either a session bean for sending messages and application callable receiving of messages, or an MDB for event style receiving of messages. These EJBs can then take full advantage of the J2EE container runtime.

An Extended Messaging port provides an abstraction to JMS resources. There are three types of ports, as follows:

- EM output port:
  - Specifies the output queue and connection factory (this is required).
– Specifies the reply-to queue (this is optional). If a sender expects a reply and the reply-to queue is left blank, a temporary queue is opened to accept the response. Reply-to queue is required for a send with deferred response, because a temporary destination cannot be left open.

– Reply-to queue is required for late-message handling.

▶ EM input port:

– Specifies the input queue and connection factory.

– Only used by Application Callable Receiver beans.

– The reply-to queue specifies where to send replies.

▶ EM extended MDB listener port:

Used when an MDB performs the role of a late response message handler.

5.2.8 Asynchronous beans

Have you ever felt the need to spawn threads to do some work asynchronously inside your servlet or EJB code, just as you are able to accomplish in normal Java client code? The answer is most likely yes. However, the J2EE specifications strongly recommend against trying to spawn other threads in the container. This J2EE constraint has a valid reason: The presence of unmanaged threads in the application server can seriously undermine the ability of the application server to ensure a stable, optimized, and scalable execution environment. Another key issue associated with application code creating threads is that the J2EE context of the application (security, local name space, and so on) does not “flow” to the newly spawned thread. So what the executed code can do is very limited, since the thread does not have access to the full and rich J2EE programming model.

The Asynchronous Bean framework resolves these two issues by providing J2EE components access to managed threads, and also allowing their J2EE contexts to be propagated to a separate thread. By using asynchronous beans, your J2EE components will be able to submit code to be run on a separate thread and asynchronously. The code will execute in a full J2EE execution context derived from your main thread of work, and thus can use the full J2EE APIs. The work will execute on threads taken from the WebSphere thread pool, and thus avoid the overhead of having to create threads on the fly. Most importantly, WebSphere has control of all these threads, allowing for better utilization of application server resources.

An asynchronous bean is a Java object or enterprise bean that can be executed asynchronously by a J2EE application, using the J2EE context of the bean’s creator. The following J2EE contexts can be inherited:
Examples of when to use asynchronous beans:

- Partition tasks so that they can run in parallel.
- Integrate non-JMS messaging middleware.
- Dynamically listen for JMS queues and topics.
- The use of background processing for performance.

The three types of asynchronous beans are as follows:

- **Work**
  
  A work object implements the `com.ibm.websphere.asynchbeans.Work` interface and runs parallel to its caller using the `WorkManager.startWork()` method. Applications implement work objects to run code blocks asynchronously. The work manager will call the `run()` method on your Work object when it spawns the thread, having already set the execution context for this thread. The work manager will call the `release()` method when it wants to terminate the thread of your work.

- **AlarmListener**
  
  An alarm is basically a timer that can be created by a J2EE application and that can be associated with a listener to invoke in case the alarm times out. The listener will be invoked on a separate thread other than the alarm creator's thread, but it will inherit the same J2EE context as the alarm creator.

  An alarm listener is an object that implements the `com.ibm.websphere.asynchbeans` interface. The `AlarmListener` interface is called when a high-speed transient alarm expires. The alarm listener's `fired()` method is called when an alarm fires. You implement any business logic that should occur inside this method.

  An alarm can be cancelled if it is a one-time event or reset if it is a repeating alarm.

- **EventListener**
  
  Event listeners are asynchronous by nature. An application can create an event listener to monitor the occurrence of a certain event. When the event occurs, the listener will be notified and will handle the event. This is a lightweight asynchronous notification mechanism for asynchronous events within a single JVM. Its main use is envisioned to be J2EE components within a single EAR signaling each other about various application asynchronous events.
Event listeners can implement any interface; however, the application that originates the event needs to know which method corresponds to the event on the listener’s interface. The event originator will do this by acquiring a “proxy” from the EventSource. Calling a method on the proxy will cause the same method to be invoked on all listeners that are registered and implement the requested interface.

The listener’s method will be executed in its own thread, but it will run under the J2EE context of the component that registered the listener itself. It will not use the J2EE context of the application that is firing the event.

**Asynchronous beans programming interfaces**

Asynchronous beans provide full support of application-controlled threading, asynchronous callbacks, scoped alarms, and subsystem monitors, and yet it is simple to use interfaces to your servlet or EJB code.

We can group the asynchronous beans interfaces in the following manner:

- WorkManager, Work, WorkEvent, WorkItem and WorkListener related
- Event and EventSource related
- AsynchScope related
- AlarmManager, Alarm and AlarmListener related
- SubsystemMonitor related

The key inheritance relationships in the Asynchronous Bean framework are illustrated in Figure 5-18.
At the heart of the APIs is the WorkManager interface, because it is the “anchor point” that allows us to create all the various asynchronous beans. A WorkManager in its simplest form is a thread pool. Its special nature allows us to use it to run Work instances asynchronously and to transfer J2EE context to those threads.

The WebSphere administrator creates WorkManagers for J2EE applications that require them. The administrator specifies the properties of the thread pool and the “sticky” context policy for any asynchronous beans using this
WorkManager. There can be many of these WorkManagers. The administrator binds each one in a unique place in JNDI. Applications will look up the WorkManager using resource references (resource-ref).

This interface includes the startWork() method, used to initiate asynchronous work.

➤ EventSources and Event

Asynchronous beans provide a generic event notification framework. An application can create generic listeners and subscribe those listeners to monitor certain events produced by an EventSource.

Several special classes in WebSphere Business Integration Server Foundation are already an EventSource, such as AlarmManager, AsynchScope, WorkManager and SubSystemMonitor. Applications interested in monitoring events fired from these sources can implement event listeners and register with the event sources.

There is also a special event source that you can use for intra-application notification. This event source is included in each enterprise application. You can look it up by using JNDI reference java:comp/websphere/ApplicationNotificationService. This intra-application event notification service makes it possible for components that belong to the same application to communicate with each other through notification. For example, an EJB could subscribe a listener, while another EJB could fire an event.

The EventSource provides “type-safe” notification. You can ask an EventSource to fire an event and target only the listeners that implement a certain interface.

Keep in mind that the intra-application notification is not valid for communications that involve multiple EARs, where instead JMS should be utilized.

➤ AsynchScope

AsynchScope is a scoping mechanism. It owns an AlarmManager and a SubSystemMonitorManager. If the scope is destroyed, then any monitors managed by its alarm manager and subsystem monitor manager are also destroyed. Properties can be stored in an AsynchScope, giving J2EE applications a way to store a non-serializable state that otherwise could not be stored in a session bean (Alarms, WorkItems, application data).

AsynchScope can have children that are also AsynchScopes. These can be useful for scoping data underneath the parent. If the parent is destroyed, then the children are destroyed also. AsynchScopes are named, and all scopes at the same level of the tree must be uniquely named. Ultimately, a WorkManager owns all AsynchScopes. This hierarchical structure is useful.
when you need to monitor complex subsystems that may have a hierarchical structure themselves.

The WorkManager can be used to create an AsynchScope. The AsynchScopes will utilize the thread pool provided by the WorkManager.

- **AlarmManager and Alarm**

  A special application of the EventSource and Listener pattern is provided by the Alarms. An application accesses an AlarmManager configured within every AsynchScope. AlarmManager provides the create() method to create a new alarm, specifying the target for the alarm, a context object for the alarm containing alarm-specific data to the listener, and when to fire the alarm.

  An alarm is a device that sits on its own thread in the WorkManager and will go off after the specified number of milliseconds. When the alarm goes off, it will call the fired() method on the listener, using the J2EE context of the alarm creator. The creator can interact with the alarm manager, reset, or cancel the alarm.

  Alarms are high performing and transient. The application that is using them needs to recreate them after server shutdown and restart. Alarms are not persistent. If you need to provide a persistent definition of time-activated tasks, the Scheduler Service should be used instead.

- **SubsystemMonitor**

  A subsystem monitor is returned to allow an application to interact with the monitor created on its behalf. This is an event source so an application can register a listener to it. An application can get ahold of a SubsystemMonitor through a SubsystemMonitorManager, which is configured within every AsynchScope.

  A subsystem monitor is basically a set of alarms. When it is created, the application tells it how often heartbeats should be expected, how many beats missed means that it is stale, and how many beats missed means it is dead. The monitor then sets up alarms to track this status. If the ping method is called, then these alarms are reset. If the alarm fires, then this means that ping has not been called and no heartbeat was received by the application for the subsystem this monitor is “watching”.

  When the number of beats for stale has elapsed without a ping, then we fire a stale event. Later, if the number of beats for dead elapse without a ping, then we fire a dead event. If a ping is received after a stale or dead notification, then we send a fresh event indicating the subsystem is alive again and carry on as normal.

  If the stale beats equal the dead beats, then no stale event is published, and we just publish a dead event. The number of dead beats should always be greater or equal to the number of stale beats. What constitutes a ping is application specific. A destroy notification is also published.
5.2.9 Scheduler and calendars

The scheduler service extends the normal J2EE services by introducing time-driven actions. These actions can be scheduled to happen only once, some time in the future, or on a recurring basis or at regular intervals. The scheduler service can also receive notifications about task activity.

Some of the advantages of using the scheduler service include:

- The scheduler service facilitates and aids in development of date- or time-based functionality.
- Scheduler tasks can be persisted in a relational database. This ensures that the task definition is retained as WebSphere Business Integration Server Foundation processes are recycled, in a single machine or cell configuration.
- The scheduler service can be workload-managed for high availability configurations, and for performance.
- Calendars are used to calculate the timing for a given scheduler task. There are out-of-box calendars that are shipped with WebSphere Business Integration Server Foundation. However, customized, business-oriented calendars can be developed to suite-specific scheduler needs.
- Scheduled actions can be EJB-based or can be triggered using JMS messages.
- The scheduler service can perform notification actions on various task activity events.

The Scheduler is useful any time you need to ensure that certain actions are taken on a regular basis. For example:

- Schedule regular backups, cleanups, night-time, or “batch” processing.
- Implement automatic, non-user driven activities, as reminders, to-do actions, etc.

Using the scheduler service

Using the scheduler service is done programmatically in your J2EE application. The Scheduler API is contained in the com.ibm.websphere.scheduler package.

To use the service do the following:

1. Configure the runtime environment using the WebSphere administration tools.
   - The scheduler configuration specifies the JNDI name where the instance will be bound, the database JNDI name, and work manager.
2. Develop the scheduler client code. This involves the following:

a. Define tasks.

Defining a task consists of creating a task information object and creating a task action object. The task information object contains generic information such as start times, repeat interval, the calendar to use, start interval, state, and persistence. It also specifies the target action and contains information specific to the target. The target action can be a JMS message or session EJB.

b. Define notifications (optional).

Various events regarding task status can be generated by the scheduler service, for example when the task is scheduled, upon successful execution of the task, upon cancelled execution, upon deleting, etc. When using the notification mechanism, you define a notification mask (specifying which events will be monitored and notifications sent), and the notification action to be performed upon the occurrence of the specified events. The action is specified by an EJB 2.0 session bean.

c. Define calendar (optional).

When creating a task information object, there is an option to specify user-defined calendar(s). This can be very useful, since different types of businesses rely on different calendars for calculations. Even within the same business, we can have different calendars. For example, national holidays differ from country to country, so the term next business day according to a given date can have a different value.

Using user-defined calendars is optional. WebSphere implements two calendars, a CRON-like calendar and a simple arithmetic calendar, that cover the majority of the needs. These calendars are used by default.

d. Find the scheduler and schedule tasks.

Programmatically, you use JNDI lookup to find the scheduler service on the application server and then schedule the task. You need to provide the BeanTaskInfo object if the task will invoke a stateless session EJB or the MessageTaskInfo if the task will send a message to a JMS destination.

The scheduler is available only to server-side components where the Scheduler daemon is running. If a scheduler resource is defined at the
node level, for example, all the application servers on that node have access to that scheduler and will run its daemon. You can selectively disable the daemon on an application server.

e. Manage scheduled tasks.

Tasks are managed programmatically by using the Scheduler API. Once a persistent task is created, it will exist and run until we (programmatically) delete it from the database. Once you have obtained the service using the JNDI, you can suspend, resume, cancel, delete, or check the status of a task. Other methods, such as finding a task or getting a task’s ID, are also implemented.

3. Add the code to your application.

Adding the code to your application is simply a matter of creating a session bean with the correct home and remote interfaces. Local name references are used for the scheduler service. You will need to include the schedulerclient.jar file in the Java classpath.

The task definition EJBs, notification EJBs, and the code that uses the scheduler with those beans are all included in your application as a part of the EAR.

**Scheduler service runtime**

An important part of the scheduler service is the wakeup daemon. It is an asynchronous bean created when the scheduler service is started.

The wakeup daemon thread wakes up periodically and reads the tasks from the database that will start in the next scheduled interval according to the scheduler service settings. It then creates alarms for each of the forthcoming tasks by using alarm threads from the assigned work manager pool.

Each alarm thread has the firetime set according to the specified task’s firetime. When the alarm thread wakes up, it tries to run the action that is associated with its task. To prevent concurrent firing of the same action, there is a locking mechanism built in. The alarm thread is aborted if the locking mechanism finds out that some other daemon has already fired the task.

### 5.2.10 Startup beans

Throughout the evolution of J2EE application design, there was always a need to do initialization or cleanup tasks before the application starts. This can be solved in different ways. In most cases, we end up without a JNDI environment for locating application resources and also a lack of security.
For this purpose, WebSphere Business Integration Server Foundation provides the startup bean service. A startup bean is a special session EJB loaded and executed by the EJB container just before an application starts. An application will not complete its startup until the startup bean has completed the execution.

The advantages of using startup beans are as follows:

- Startup beans run with full security context.
- Startup beans run within WebSphere’s name space. Therefore it uses JNDI to find and use other resources.
- If we compare it to a servlet, the servlet's init() method is often used as a startup mechanism for starting initialization tasks.
- The startup bean also provides a method that executes upon the application’s shutdown.

Startup beans can be used in various situations, for example:

- To check the connections of databases or legacy systems before starting the application.
- To load data into the cache.
- To “warm up” entity beans by executing their finder method and storing the EJB handles for future use. This is also known as “commit option A” EJB cache and can be even more effective together with Async Beans.
- To asynchronously populate a cache and start an application. If an application uses a large cache of objects, but can still work without the cache, it is not necessary for the application to wait until the full cache is built. With a Startup Bean we can initiate an Async Bean, which populates the cache and starts the application.
- To implement independent fast logging. Asynchronous Beans can be used for writing application logs to the database, and Startup Beans serve as a fast bootstrap mechanism.
- To create a task and schedule it with the Scheduler service before an application starts.

Using startup beans
A startup bean is a user-defined EJB 2.0 session bean placed in the application. The startup bean can be either stateful or stateless. If it is stateful, the same instance is used for start and stop. Otherwise, two instances are created. To successfully compile the application and generate deploy code, startupbean.jar must be added to the Java build path.

It is mandatory to use the following home and remote interfaces:
- EJB home interface must be:
  com.ibm.websphere.startupservice.AppStartUpHome

- EJB remote interface must be:
  com.ibm.websphere.startupservice.AppStartUp

This assures that the WebSphere runtime will recognize the bean as a startup bean and proceed accordingly.

An application can have more than one startup bean. In this situation, priorities (defined as the wasStartupPriority environment properties) must be defined for the beans. For this purpose, an environment property must be defined on each startup bean named name of this priority must be and its type must be java.lang.Integer.

The remote interface defines the following methods that must be implemented:

- boolean start()
  
  This method is called before an application starts. From this method we call all the user-defined initialization code we want to execute.

  The start() method returns true when everything goes right, indicating that the application will start. Returning false or throwing any exceptions indicates that application start will be aborted.

  Because the application will not start before this method completes successfully, consider the use of asynchronous beans if the initialization task is not necessary to complete fully prior to the application's start.

- void stop()

  This method is called before the application stops. More exactly, it is called after you initiate the stop of the application and all the requests have been served. This method can call all the user-defined “cleanup” code we want to execute.

  Exceptions thrown by this method will be logged but ignored, and the application will continue to stop.

### Startup service runtime flow

The following steps represent different states in the life cycle of a startup bean in runtime.

1. Application startup

   After the application’s EAR is loaded, service looks for all the startup beans, finding them according to the home interface. Then, for each startup bean it looks for the presence of wasStartupPriority as a property defined for the startup EJB in the deployment descriptor. If there is no priority property...
defined or it is the wrong type, it would be set to the lowest possible value for the integer.

Startup beans are ordered according to priority, and the startup service takes the bean with the highest priority from the queue. It checks its transactional properties on start() and stop() methods. If a TX_MANDATORY exception is thrown, startup of the application is aborted. Then it finds the actual bean using JNDI and runs its start() method. If there was no problem with the lookup and the start() method returned true upon its execution, the startup service stores the bean handle and proceeds to the next startup bean in the ordered queue as long there are any startup beans left. When it finishes successfully with all the startup beans, WebSphere starts the application and its JMS, IIOP, and HTTP listeners.

**Important:** Startup service waits until the start() method finishes its execution. So, if there is a demanding task triggered by the start() method, it could take some time before applications actually start. In those situations, it is recommended that you use Asynchronous Beans to do the task and therefore finish with the start() method as soon as possible.

2. Application shutdown

In the application shutdown process, right after the containers stopped serving the requests, startup beans are taken one by one in the reverse order of precedence regarding the application’s startup. For each bean, stop() is executed. Any exception thrown at this stage is ignored, and startup service continues with the next bean. Finally, it shuts down the application and the listeners.

3. EJB module re-start

Let us suppose we have an application with more than one EJB module and each contains startup bean(s). If one of the EJB modules is stopped and then restarted, the startup service will consider it an application start. Therefore, it will execute the startup beans that are part of the restarted EJB module.

4. Application server crash

If the application server crashes, the startup bean stop() method may not be executed. When restarting the server, the start() method will be executed normally.

### 5.2.11 Object pooling

The Object Pool service can be used to improve the performance of multi-threaded applications that frequently use a common complex object. To do this, a Java application will have its own pool of objects that are already
instantiated and waiting to be used. When the application needs a new object, it will simply “fetch” one from the pool and return the object to the pool when it is no longer required (instead of being destroyed). This returned object can then be later reused by another thread.

These object pools are not intended for pooling JDBC connections or JMS connections and sessions. The server tools provide specialized mechanisms for dealing with those types of objects. These object pools are designed for pooling application-defined objects or Developer Kit types such as Vectors or HashMaps.

Object pooling is implemented within the J2EE application container and is closely coupled with the rest of the services in the container. It uses the PMI and the tracing infrastructure, allowing you to monitor what is happening with the object pool in the runtime. If the provided object pool implementation does not suit your application, you can create your own object pool implementation, which is then managed by WebSphere’s object pool manager.

Generally, you should consider using object pools if your application consists of many threads that frequently use a common complex object. The performance benefits of using object pools should be carefully considered. The majority of applications do not need to use object pools to obtain adequate performance, but those who need it can gain significant performance improvements. When considering whether to use object pools or not, you should think about the following two issues:

- Size and complexity of the object
  Using object pools with simple object types can slow down application performance unless the usage frequency is so high that you can achieve improvement over normal JVM garbage collection.

- Object usage frequency
  Logically, more frequent use of some objects qualify them better to be used with object pooling.

**Object Pools API**

Using object pools requires that you include the objectpool.jar in the project’s Java classpath. The Object Pools API is contained in the com.ibm.websphere.asynchbeans.pool package. It contains the following interfaces.

- Object pool manager
  This interface describes the object pool manager, which is a factory for object pools. You can define one or more managers and register them in the WebSphere JNDI name space using administrative console.

  The object pool manager interface defines two methods:
– createFastPool(): This method returns an unsynchronized object pool.
– getPool(): This method returns a synchronized object pool.

Object pool

The object pool is a pool of reusable objects of the same type. WebSphere's default object pool implementation implements this interface. Using the default implementation, any Java object that has a public default constructor can be pooled.

This interface defines two methods:
– getObject(): Retrieves an object from the object pool
– returnObject(): Returns an object that is no longer needed to the object pool

Custom object pool

Custom object pools are used when WebSphere's object pool implementation does not suit the needs of your application, for example, if you need a notification about taking and returning objects from the pool. Since the default WebSphere object pool implementation does not provide this, you can use this interface to create your object pool implementation. The custom ObjectPool interface extends the ObjectPool interface with additional methods that you need to implement beside the methods inherited from the ObjectPool interface. The methods defined in the custom ObjectPool interface are:
– flushPool(): Called when the system needs memory, so it asks you to clean all the idle objects.
– setProperties(): Called when the custom object pool is constructed. A map of the supplied parameters is supplied. These are the additional optional parameters added to the custom object pool configuration.

Each custom pool must be registered within an object pool manager.

PoolableObject

When pooling the object of your own defined Java class types using the default object pool implementation, you might need to have some initialization or cleanup code in the object when the object is taken from the pool or returned to the pool. Such an object needs to implement the PoolableObject interface with the following methods:
– init() is called when the object is about to be reused from the pool.
– returned() is called when the object is returned to the pool.

The default pool implementation checks if the object being pooled implements this interface. If it does, then it will call the init() method when getting the object from the pool and it will call the returned() method when the returning the object.
Using object pools
The following steps must be done in an application when programming for object pools.

- Find the object pool manager.
  The object pool manager is exposed in the JNDI name space. The manager has to be created in the runtime using the Administrative Console. Programmatically, you do a lookup on a context to find an object manager.

- Get an object pool for specified class.
  Ask the object pool manager to get an object pool for the class you want to be pooled. This is done by calling the `getPool()` method on the object pool manager, which will then create an object pool for the given class and return it to your control.
  The `createFastPool()` method can also be called. In this case, the object pool manager creates and returns an object pool that is not a thread-safe, unsynchronized object pool.

- Get an object from the object pool.
  After we have the object pool created, we can get an object of the specified class by calling the `getObject()` method of the object pool.

- Return the object to the object pool.
  After the object is not needed by the application anymore, return the object back to the object pool by calling `returnObject()` method of the object pool.

Runtime environment
The object pool service is configured using the WebSphere Administrative Console. First you have to configure an object pool manager and then you can create object pools under the object pool manager.

WebSphere offers two types of object pools: Synchronized and unsynchronized.

- The synchronized type of pool is thread safe. A synchronized pool is useful for applications where little contention exists for objects in the pool. A synchronized pool can be shared across threads, which means more efficient use of pooled objects, because there are fewer idle instances of objects in the pool at any given time. In addition, a single shared pool is easier to manage than multiple pools.
  Underneath, the synchronized pool is in fact a wrapped unsynchronized pool.

- Unsynchronized pools are not thread safe. Applications can only use them within one thread. They are faster than the synchronized, so if your application uses an object pool in one thread, we recommend the use of unsynchronized pool.
5.2.12 Internationalization

An application that can present information to users according to regional cultural conventions is said to be internationalized. The application can be configured to interact with users from different localities in culturally appropriate ways. In an internationalized application, a user in one region sees error messages, output, and interface elements in the requested language. Date and time formats, as well as currencies, are presented appropriately for users in the specified region. A user in another region sees output in the conventional language or format for that region.

Internationalization of an application is driven by two variables: The time zone and the locale. The time zone indicates how to compute the local time as an offset from a standard time, such as Greenwich Mean Time (GMT). The locale is a collection of information about language, currency, and the conventions for presenting such information as dates. In a localized application, the locale also indicates the message catalog from which an application is to retrieve message strings. A time zone can cover many locales, and a single locale can span time zones. With both time zone and locale, the date, time, currency, and language for users in a specific region can be determined.

The Internationalization service provides a mechanism for propagating locale and time zone information from clients to servers and between server components. This information can be used by server application components to customize the results according to the client locale and time zone. The transfer of the locale and time zone information is done transparently by the Internationalization (i18n) service.

Consider the following situations:

- Client and server processes can run on computers that have different locale settings.
- Client and server processes can execute in geographical locations having different time zones.

The conventional solution for solving locale and time zone mismatch problems is to pass one or more extra parameters on all business methods necessary for conveying either the client’s locale or time zone to the server. Although the technique is simple, it has serious limitations:

- Parameter lists become longer.
- Extra parameters may need to be added to all methods to pass the information through the call chain even if they do not use the information.
- Adding extra parameters to deployed applications is inherently error-prone.
It may be impossible to add parameters to the applications that are not easily modified, such as legacy applications.

The WebSphere internationalization service solves locale and time zone mismatch problems without the traditional limitations. The locale list and time zone are managed as a unit referred to as an internationalization context. The internationalization service manages the distribution of internationalization context across the various components of enterprise applications, including Java client applications, EJBs, JSPs, and servlets. The server-side components can use the internationalization context API to access the distributed internationalization context and then localize computations to the locale or time zone of the client-side components.

The service associates an internationalization context with each thread of execution in an application. When a client-side program invokes a remote business method, the Internationalization service obtains the context associated with the current thread and attaches it to the outgoing request. On the server side, the Internationalization service detaches the caller's context from the incoming request and associates it with the remote business method thread. The service propagates this context on subsequent remote business method invocations to pass the context of the original request down the call chain.

This support is extended to Web services applications using the same concepts. The internationalization context is transparently propagated over the requests originating from the J2EE Web services clients. On an outgoing Web services request, the service creates a SOAP header block containing the invocation context associated to the current thread, and this SOAP representation is then inserted into the outgoing request.

When a request arrives at the server side, the service scopes the propagated internationalization context, referred to as the caller context, to the invocation of the stateless session bean enabled as a Web service. It also scopes an invocation context to the invocation in case the internationalization context management policies are CMI and run as Caller.

To perform the localizations, Web services-enabled EJBs will obtain the elements of either context using the internationalization context API and utilize them within locale- or time zone-sensitive operations.

The internationalization service is bound to the JNDI. Then invocation internationalization context is retrieved from the UserInternationalization interface. On the Web services client, we can set the locale and time zone to the invocation internationalization context. This context is passed to the server-side components and retrieved as the caller internationalization context.
Using internationalization

Every server application component that runs on WebSphere Business Integration Server Foundation has one internationalization type setting. By this setting, the server container decides that the invocation internationalization context is managed by the application component or by the hosting J2EE container.

The server application components can be deployed to use one of the following types of internationalization context management:

- **Application-managed internationalization (AMI)**
  
  Under the AMI deployment policy, component developers assume complete control over the invocation internationalization context. AMI components can use the internationalization context API to programmatically set invocation context elements.

- **Container-managed internationalization (CMI)**
  
  Under the CMI, the Internationalization service collaborates with the Web and EJB containers to set the invocation internationalization context for the servlets and EJBs. The service sets invocation context according to the container internationalization attribute of the policy associated with a servlet (service method) or an EJB business method.

The applicable settings of the internationalization type depend on the component type. The following list shows the applicable settings for each type of components:

- Servlets, session beans, and message-driven beans can be deployed as AMI or CMI, but not both. CMI is the default.

- The internationalization type of the entity beans is CMI and it cannot be configured.

- EJB client applications do not have an internationalization type setting, but are implicitly AMI.

To develop the application that uses the internationalization context API, you have to put the i18nctx.jar file in your classpath.

The applications use the API to access and manage internationalization context. Three interfaces are provided by the com.ibm.websphere.i18n.context package for this purpose:

- com.ibm.websphere.i18n.context.UserInternationalization
- com.ibm.websphere.i18n.context.Internationalization
- com.ibm.websphere.i18n.context.InvocationInternationalization
5.2.13 Shared work area

In the process of developing software applications, the need to pass data between application components is often a fundamental requirement. A significant part of this is contextual information such as security information, transaction context, locale information, and business state data. When developing distributed applications, potentially spanning multiple logical and physical tiers, the process of passing context data around can be quite challenging. As a particular user request flows from the originating client through one or more application tiers, the information has to be always available to the current component for processing. Even if the information is not needed for a particular component, it must be preserved and propagated to components further down the process flow in case they need the information to complete their jobs.

In a typical application, a browser-initiated request might call a servlet, which then calls a session EJB that calls multiple entity EJBs. In this case, the servlet, session EJB, and entity EJB would all need access to security information, transactional information, and possibly other business information. Fortunately, with J2EE, the underlying protocol RMI/IIOP has the ability to implicitly carry private context information on the thread of execution without the developer having to write any code. This is how security and transaction contexts are propagated from an EJB client (a servlet in this example) to each EJB that it calls, as well as to each EJB that those EJBs call, whether the EJBs are on the same physical tier or separate ones.

This takes care of a large part of the “standard” contextual data that needs to flow from tier to tier, but what about the business data that might be necessary or at least convenient to have available to every component? For instance, when a user initiates a session it might be beneficial to look up some profile information and make that available to all the components that are called by the application.

There are a couple of alternatives to achieve this. One would be to add a custom profile key or object to every method call so the providing component has access to the profile data. This would have to be done even in cases where the component did not need the data, but other components further down the line might need it to complete their functions. This alternative would have several negative side effects, including making every method call more complex, increasing maintenance complexity, and reducing the reuse potential of many EJBs. Clever design and/or use of a custom framework might reduce the severity of these side effects, but they would still pose significant challenges. In the case where components have been purchased from a third party, adapting pre-written components in this fashion might be literally impossible without participation from the component vendor.
A better alternative would be to have the underlying middleware “plumbing” carry the client profile information the same way that it carries security and transactional information. This provides a much simpler programming model, as well as much more flexibility and maintainability.

**Work area service**

The shared work area service provides a simple, flexible solution to the problem outlined above. Using this service, developers can easily create a work area, insert data into it, and make remote invocations to EJBs. The work area will be propagated with each remote method invocation in the same way that the security and transaction contexts are. The receiving component may use the data, ignore it, or add more to it. If the receiving component calls methods on other components, the work area data will flow along to those components as well. When the original client is done with the work area, it terminates it.

A work area is defined as a set of properties, each of which contains a key (that uniquely identifies the element), a value (the actual data that needs to be propagated), and a mode (which indicates whether the data can be modified or deleted by downstream components). Each piece of data can be accessed separately, so if the type or length of the data changes, only those methods that actually use that particular piece of data are affected by the change. Work areas can also be nested to provide control over data visibility and even override specific properties.

There are two prime considerations in deciding whether to pass specific information explicitly as an argument or implicitly by using a work area:

- **Pervasiveness**
  
  How much of the given information is used in your application overall? Is it used in a majority of the methods in an application? The Shared Work Area service is best used to store data that is required by many parts of the application. By using shared work areas, only those methods that actually use the data would have to handle it. Because a shared work area is not explicitly passed, there are no extra properties for methods that do not need the data. In this sense, a work area is similar to the security or transaction context in WebSphere Application Server, but in contrast to those two, it is generalized and exposed for use in your application.

- **Size**
  
  Is it reasonable to send the information even when it will not be used? The size of information used with shared work areas is important from a performance point of view. Since we cannot always predict and control the size programmatically, WebSphere’s implementation has built in some configurable limitations for the size of information stored in a shared work area.
When using shared work areas, the data can only be passed one way, from the caller to the remote method. If the remote method makes changes to the data, the changes are never seen by the caller. A remote method can be another servlet or a method in the remote interface of an EJB. Even if the servlet or bean resides in the same JVM, the call is still considered remote.

When using shared work areas, you need to include the acwa.jar and distexcep.jar files in the application's Java classpath. The Work Area API is contained in the com.ibm.websphere.workarea package. It contains the following interfaces:

- UserWorkArea: Defines the work area class
- PropertyModeType: Allows you define permissions for each property in a work area

For a particular work area, only the originator thread (the thread that created and "owns" the given work area) can modify, remove, or change the mode of properties created in that work area.

The following steps have to be done in your application when programming for work areas:

1. Find the shared work area service.
   - The work area object is exposed in the JNDI name space. By default it is enabled, and you can access it by doing a lookup on a context to find the Shared Work Area service.
   - The Shared Work Area service is available within the WebSphere J2EE server environment and also within the J2EE client container. In both cases the same procedure is used to get the service.

2. Create a work area.
   - When you have the Shared Work Area service you can create and associate a new work area for the current thread of execution. This is done by invoking the begin() method. If there are already existing work areas associated with the thread, a nested work area will be created.

3. Use the work area.
   - The set() method is used to set user-defined properties. You can put whatever you want in the work area value as long as your user-defined class implements the java.io.Serializable interface. Each property is represented by a name, value, and mode triplet.
   - To get a property from a work area, invoke a get() method supplying the property name.
   - To modify a property, invoke the set() method again with the same property name. You can only do that in the originator thread (the thread that created
the work area). It is possible to modify a property of the work area in a non-originator thread by creating a nested work area with the same property if the original property allows it to be overridden. You can also change the mode of a property, but the same circumstances apply.

4. Terminate the work area.

When you no longer need a work area, you should terminate it. Again, you can only terminate a work area that was created in the current thread. If the current thread just has a work area inherited from a remote caller, then it cannot terminate it. It can only terminate its work areas and nested work areas.

**Nested work areas**

A nested work area is a work area created “over” another work area. Let us say you have a thread that got a work area from a remote caller. When your thread creates another work area, we call the new work area to be nested. When using the work area in the current thread, you will see all the properties. The originating work area, plus the properties from the newly created nested work area, will appear as one flat work area. The remote caller will not be aware of any nested work areas, since the originating thread will not see anything from the newly created nested work area.

In contrast, when you make a remote call to another thread from the current thread (the one that inherited a work area and then created a nested work area over), the called thread will get both work areas, and again it will appear like one flat work area.

Nested work areas are also used to override the properties from the originating work areas. Since you cannot change properties in a work area if you are not the originator who created the work area, the only chance is to create a nested work area with the same property. And in the nested work area, you can only create the same property if the inherited property permits it, which it can do if its mode is not any of the read-only modes.

**Work Area Partition service**

The Work Area Partition service is an extension of work area service that allows users to create multiple customer work areas. A WorkAreaPartition can be thought of in the same way as the UserWorkArea and has the same API as the UserWorkArea. Any user that currently uses the work area service and the UserWorkArea partition can continue using it in the same manner. A WorkAreaPartition differs from the UserWorkArea in a number of ways.

- The UserWorkArea is publicly available, through "java:comp/websphere/UserWorkArea", thus allowing multiple users on the same thread to access the context in the WorkArea. A work area partition is
created by a certain user and is only known to that user, unless of course that user makes its partition publicly known. The Work Area Partition service does not strictly enforce that a partition is accessed by the partition creator. It can be accessed by anyone who knows the certain partition name. Users can choose to publish a certain partition name or not. On the other side, Work Area Partition service will try to hide a partition as much as possible; actually, it does not allow a person to determine or query all the names of partitions that have been created.

- A WorkAreaPartition can be configured to allow for bidirectional propagation of its context. That is to say, changes made to a WorkArea's context by a downstream process will be propagated back upstream to the originator of that WorkArea.

- The serialization and deserialization of attributes are automatically performed by the work area services. With the Work Area Partition service, you can enhance performance by deferring the deserialization using the deferred attribute serialization cache service. With this property specified (using the administrative console or the API), attributes set into the work area service are not automatically serialized during the set operation. Rather, the work area stored a reference to the attribute as a cache. When a get operation is performed on an attribute, the reference to the object is returned to the requester and no deserialization is performed.

- If a client has multiple different work area partitions when it makes remote invocation to server side, the information included in each partition on the client thread propagates to the application server. Which information (context) will be demarshalled at server side depends on the appropriate partition name defined both at client and server side. It means that only the context associated with a partition that is resident on both the client and server is demarshalled. The information (context) associated with a partition that does not reside on server side is still there, but will not accessible; it will be propagated to a different server during the next remote invocation.

The WorkAreaPartitionManager.createWorkAreaPartition() method can only be used from J2EE clients. Administrators can use the administrative console to create a work area partition on the server side. On the server side a work area partition must be created during server startup because each partition needs to be registered with the appropriate Web and EJB collaborators before the server is started. When creating the work area partition, there are some configuration properties available, which are defined as described in the following sections.
5.3 Technology preview: Common Event Infrastructure

The Common Event Infrastructure (CEI) is a set of modular components that provide simple event management. CEI has been designed as a core technology that will be integrated into many other IBM products in the medium term. WebSphere Business Integration Server Foundation is one of the first application platforms to provide an implementation of this core technology.

The fundamental concept of the CEI is that applications or middleware components create events whenever they perform some processing that could be of relevance to an external application. The event contains information relating to event identification, timing, and other details.

The component creating the event object is called the event source. The event object is passed to the event infrastructure, whose role is to forward the event on to event consumers. These are other applications that have expressed an interest in the event. The event infrastructure may also store the event object in a database for later retrieval.

Figure 5-19 shows the activities that are supported by the CEI.

Note: The Technology Preview version of CEI does not support the event distribution. It is only going to be available in the full version of this technology in a future release of WebSphere Business Integration Server Foundation.
The standard structure used for the event objects is the Common Base Event (CBE), which is part of the IBM Autonomic Computing Toolkit. The CBE standard defines a set of common fields, the values they can take, and the meaning of these values. Further details of the CBE and the IBM Autonomic Toolkit can be found at:

- “Standardize messages with the Common Base Event model”, IBM developerWorks article:
  
- “Specification: Common Base Event”, IBM developerWorks:
  

WebSphere Business Integration Server Foundation provides support for both the CEI client and server components.

- Client support is provided by three mechanisms. CBEs can be created:
  - Explicitly, with a Java API
  - By configuring deployment descriptors

  **Note:** The deployment descriptor method is not available in the technology preview, and so is not covered in detail by this publication.

  - By configuring activities within a BPEL4WS process

    These are passed to the State Observer plug-in (SOP), which uses Business Context Data Services (BCDS) to populate the CBE.

- Server components include:
  - A database to store the CBEs

  **Note:** The only supported database for the technology preview is Cloudscape.

  - An application to handle the incoming events

    The application can store the events in a database, forward the events to another CEI server, or do both.

    **Note:** The technology preview only supports a single server instance of the CEI application.

  - A CBE viewer to examine the CBEs in the database
WebSphere Business Integration Message Broker V5

WebSphere Business Integration Message Broker enables information, packaged as messages, to flow between different business applications. This broker product extends the messaging capabilities of WebSphere MQ by adding message transformation and routing features.

For those familiar with the previous IBM messaging products, the functions of WebSphere MQ Integrator and WebSphere MQ Event Broker have merged into one product consisting of:

- WebSphere Business Integration Message Broker, providing broker functionality for point-to-point messaging
- WebSphere Business Integration Event Broker, providing publish/subscribe functionality

This paper will help you understand what WebSphere Business Integration Message Broker is and what it does. It will only touch on the highlights of the product. For more detailed information, see the references listed in “For more information” on page cccxiv.
6.1 Product overview

WebSphere Business Integration Message Broker consists of the following installable components (see Figure 6-1):

- **WebSphere Business Integration Event Broker**
  
  The Event Broker provides publish/subscribe capabilities, including content and topic-based message routing. If you only require the Event Broker features, you can install this component without installing the full Message Broker component.

- **WebSphere Business Integration Message Broker**
  
  The Message Broker includes the capabilities of the Event Broker. In addition it provides a broad range of broker functionality including message transformation capabilities, content dependent message routing, and database integration for logging and message enrichment.

- **Message Brokers Toolkit for WebSphere Studio**
  
  The third major component is the Message Brokers Toolkit. This Eclipse-based Workbench tool provides both the development environment and the broker administration environment. The Message Brokers Toolkit has the same look and feel as the whole line of WebSphere Studio integrated development products.

![Figure 6-1  WebSphere Business Integration Message Brokers](image-url)

- **New Era/Sybase rules engine**
  - Includes Message Broker
    - Rules and Formatting Engine from New Era of Networks

- **Transformation, Routing and Data Integration**
  - includes all Event Broker functionality
    - content and table-driven routing capabilities
    - powerful engine for message/data transformation and database integration
    - XSLT engine for XML transformation
    - message filtering and message warehousing
    - Web Services (HTTP/SOAP) protocol support

- **Multi-protocol event switch**
  - point-to-point, publish/subscribe and multicast models
    - supports MQSeries, Mobile, Telemetry, Real-time and Reliable IP Multicast protocols
    - extensible plug-in architecture
For customers that have a continuing need for the New Era of Networks Rules and Formatters nodes, the Rules & Formatter Extension is available. It is supplied as two additional installable units: a runtime unit and a tooling unit. The runtime unit is required by a broker to properly manage messages and message flows operating in the NEONMSG domain. The tooling unit is needed on machines that are expected to support the development and maintenance of formats and rules through the Formatter and Rules GUIs.

Brokers can run on a number of different platforms including Windows, AIX, Linux, HP-UX, Solaris, and z/OS. The WebSphere Studio suite is only available for Windows 2000 and XP.

WebSphere Business Integration Message Broker requires IBM DB2 and WebSphere MQ to be installed for use by Message Broker components.

### 6.2 Architecture overview

The purpose of a broker is to take incoming messages from applications and perform some action on them. The following are examples of actions that might be taken in the broker:

- Route messages to one or more of many destinations
- Transform messages to an alternative representation
- Perform message aggregation, decomposing messages into multiple messages and sending them to their destination, then recomposing the responses into one message to return to the user
- Interact with an external repository to augment a message or store it
- Invoke Web services to retrieve data
- Respond to events or errors
- Provide content and topic-based message routing using the publish/subscribe model

WebSphere Business Integration Message Broker provides both the runtime and development environment necessary to provide broker functionality. Actions taken on messages are done by message flows executing in the runtime component. As messages arrive from applications over a supported transport, they are processed by the appropriate message flow, then sent on to their destination.

Figure 6-2 on page ccxc illustrates the basic runtime and development architecture.
Figure 6-2 WebSphere Business Integration Message Broker architecture

Figure 6-2 is explained below:

1. The primary runtime component is the broker. Brokers contain a number of execution groups, which are processes in which message flows are run. Each broker uses a database to store the information it needs to process messages at runtime.

2. Messages are processed by message flows. Message flows are developed to provide specific functionality by wiring a series of nodes together. Each node has a specific job to do within the scheme of the message flow. Nodes for input and output are designed to take the messages from specific transport types. Other nodes can perform computations, message enhancement, or make routing decisions.

3. Messages must have a defined structure which is known and agreed to by the sender and the receiver. In order for the broker to process messages, it must also understand their format. A message set contains message definition files
that describe the messages. As the messages are processed, message flows apply the appropriate message definitions to them at various stages during its progress along the flow.

4. The Message Brokers Toolkit for WebSphere Studio provides an integrated development environment for message flow development and runtime administration. Message flows and message sets can be developed using the Message Brokers Toolkit Workbench, then packaged for execution and deployed to the runtime environment via a connection established between the Workbench and the Configuration Manager.

5. The Configuration Manager coordinates all activity (for example, changes to a message set) between the Workbench and brokers within its domain. Brokers are grouped into broker domains. Each domain is coordinated by a Configuration Manager. It uses a database as a repository to store information relating to its broker domain.

6. If you have applications that use the publish/subscribe services of a broker, you can apply an additional level of security to the topics on which messages are published and subscribed. This additional security, known as topic-based security, is managed by the User Name Server. It provides administrative control over who can publish and who can subscribe.

7. Transport support facilities provide the interface between the client applications and the message flows.

WebSphere MQ queue manager(s) provide the underlying transport infrastructure for the WebSphere Business Integration Message Broker. IBM WebSphere MQ messaging is used between the Workbench, the Configuration Manager, and the brokers. WebSphere MQ is also one of the transports supported for communication between applications and brokers as well.

### 6.3 Transport support

WebSphere Business Integration Message Broker provides function and transport capabilities which support and facilitate enterprise level business integration.

End-user applications can connect to the broker using one of the following transports:

- WebSphere MQ clients connect using the WebSphere MQ Enterprise Transport.
- WebSphere MQ Everyplace clients (pervasive devices) connect using the WebSphere MQ Mobile Transport.
- Multicast JMS clients connect using the WebSphere MQ Multicast Transport.
- Real-time JMS clients (direct TCP/IP) connect using the WebSphere MQ Real-time Transport.
- SCADA clients (remote devices) connect using the WebSphere MQ Telemetry Transport.
- Web services clients (HTTP) connect using the WebSphere MQ Web Services Transport (Message Broker only), allowing message flows to be invoked as Web services.

Figure 6-3 summarizes the transport options.

**Figure 6-3 Message Brokers and the Integration Bus**

### 6.4 Runtime environment

Figure 6-4 on page ccxciii shows an overview of the runtime environment.
6.4.1 Configuration Manager

The Configuration Manager maintains the broker domain configuration. It provides the interface between the Workbench, the configuration repository and an executing set of brokers. The Configuration Manager is required to deploy message flows, message sets and domain configuration information to a broker.

The Configuration Manager has the following primary functions:

- Maintains configuration details in the configuration repository, a set of DB2 database tables that provide a central record of the broker domain components
- Deploys WebSphere Business Integration Message Broker resources from the Workbench to the broker domain
- Reports to the Workbench on the results of deployment operations and the status of the brokers

It communicates with the other components in the broker domain via WebSphere MQ.
For each broker domain you must create and start one (and only one) Configuration Manager. The Configuration Manager can only be created on Windows systems and uses a JDBC connection to a DB2 database for its configuration repository.

The Configuration Manager is created using `mqsicreateconfigmgr` from the Window's command prompt, or using the create a default configuration hyperlink from the Toolkit Getting Started page.

As part of the Configuration Manager creation process, the repository tables are created in the DB2 database and a set of WebSphere MQ queues are defined on the queue manager which hosts the Configuration Manager. These queue names all begin with SYSTEM.BROKER. The rest of the name will indicate the purpose of the queue and it will be unique. The hosting queue manager must be on the same Windows machine as the Configuration Manager.

A single server connection channel (SYSTEM.BKR.CONFIG) is created to allow the Workbench to communicate with the Configuration Manager. You will need to manually create sender/receiver channels to brokers that are hosted by remote queue managers to allow you to administer them from the local Configuration Manager. Alternatively, you could put the all the broker queue managers in a WebSphere MQ cluster with the Configuration Manager queue manager.

Message flow development can be done in the Message Brokers Toolkit without an active connection to a Configuration Manager, but a connection must be available to be able to deploy and test message flows on a broker.

### 6.4.2 Broker

A broker is the named resource that executes the business logic defined in the message flows; see Figure 6-5 on page ccxciv for a broker overview. Applications send and receive messages to and from a broker using WebSphere MQ queues, MQe, Web services or HTTP as the methods of communication. The messages used in a broker are usually defined in a message set.

More than one broker can be defined per broker domain, either on the same or on a different physical system. If you plan to use the publish/subscribe service, you can connect a number of brokers together into a collective, which allows optimization of the subscriber connections.

Brokers contain execution groups that run as separate operating system processes, providing an isolated runtime environment for a set of deployed message flows. Flows running within an execution group are threads. By default, only one execution group is defined, but more can be easily added.
Message sets and message flows are packed into a broker archive file (BAR file) for deployment. You can logically group message flows and message sets by placing them in different BAR files. Several different BAR files can be assigned to an execution group, although using different execution groups provides better application isolation. A particular message flow can be configured to run more than one instance of itself inside the execution group. For even more isolation between applications, you can use different brokers.

Each broker requires:

- A set of tables in a database to hold the broker’s local data. This set of tables is accessed through an ODBC connection. They can be created using a number of database products depending on the broker platform:
  - DB2 UDB
  - Microsoft SQL Server (Windows only)
  - Oracle
  - Sybase

- A WebSphere MQ queue manager. A broker can share the queue manager hosting either the Configuration Manager or the optional User Name Server,
or both. However, since the broker uses a set of predetermined queue names, brokers cannot share a queue manager with another broker.

- A set of named queues on the queue manager associated with the broker domain. These are created automatically when the `mqsc createbroker` command is issued.

Each broker instance has an assigned, permanent and fixed name. This is the broker instance name. This name, which is similar to the static identifier assigned to a database before it is created, is used to distinguish tables pertaining to one broker from other tables where multiple brokers have been set up using the same database.

Creating a broker on the target execution platform does not itself update the Configuration repository; you need to create a reference to it using the Message Brokers Toolkit. Once this is done and the broker is deployed, message flows and sets can be assigned to the broker. Brokers provide information, in the form of published event messages, in response to changes; these can be used by the Broker Administration perspective to update the management agents as to the status of the broker components.

### 6.5 Message flow applications

Message flows provide broker functionality driven by business rules. They can route and transform messages, filter messages (topic-based or content-based), and have database capabilities for enrichment of the messages or for warehousing the messages. WebSphere Business Integration Message Broker also provides a framework for extending the functionality with plug-ins to user-written or third-party solutions for specific requirements.

Message flows define a set of operations that are performed on a message when it is received by a broker. A flow consists of a number of message flow nodes, which are wired together to allow messages to be processed. Each node in a flow represents a processing step. A node has terminals for input, output or both.

Looking at the bigger picture, the message flow is the intermediary between two or more external applications. The total solution is going to be based on one of two communication models:

- Point-to-point
- Publish/subscribe

In a point-to-point model, a message flow similar to the one shown in Figure 6-6 on page ccxcvii takes a message from a client and delivers copies, some transformed, to any number of back-end servers for processing. So as a
message passes through a message flow, it is transformed and routed according to the nodes it encounters.

![Diagram of message flow](image)

**Figure 6-6  WebSphere Business Integration Message Brokers message flows**

In a publish/subscribe model, shown in Figure 6-7, applications are not tied to particular partners, so they use messages that are more flexible in terms of their destinations. For example, messages are published about a particular topic as opposed to a particular application. Content-based publish/subscribe allows subscribers to specify an ESQL filter expression, which is then evaluated against each publication matching a subscription request topic.

Publications are also classified as either state-based (retained) or event-based (not retained). Retained publications are kept after being published, allowing late subscribers to obtain the most recent information.

![Diagram of publish/subscribe model](image)

**Figure 6-7  WebSphere Business Integration Message Brokers with publish/subscribe**

### 6.5.1 Building message flows

The Message Brokers Toolkit is used to build flows by placing nodes chosen from the node palette and connecting the terminals. Message flows are contained within *message flow projects*. Message flow projects contain the resources which make up one or more message flows, including Extended Structured Query
Language (ESQL) modules and message mappings. Figure 6-8 shows a simple message flow opened in the message flow editor in the Message Brokers Toolkit.

![Example of a message flow](image)

*Figure 6-8  Example of a message flow*

Message flow nodes are configured through their properties. The properties that are available depend on the type of the node. For example, the MQOutput node has a mandatory property for the queue name and an optional property for the queue manager name. For the MQInput node, you would specify the queue name, the type of message, and other relevant options (see Figure 6-9 on page ccxcix).
Some nodes can process messages using ESQL statements. ESQL is an interpreted language based on Structured Query Language (SQL) that supports the manipulation of data within a message flow. ESQL can be used to develop complex business logic, but it is more commonly used to perform simple operations on messages and database contents. The language includes looping and branching capabilities. ESQL is most often used in Compute nodes (see Example 6-1) and Filter nodes.

**Example 6-1  ESQL module for a Compute node**

```sql
CREATE COMPUTE MODULE Propagate_multiple_messages
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
  DECLARE I INTEGER;
  SET I=1;
  WHILE I<=Environment.Variables.NoPassengers DO
    SET OutputRoot = InputRoot;
    SET OutputRoot.XML = NULL;
    SET OutputRoot.XML.PassengerReservationResponse.ListOfConfirmations.Confirmation = InputRoot.XML.Reservation.ListOfPassengers.PassengerDetails[I];
    PROPAGATE;
    SET I=I+1;
  END;
END;
```
Other nodes provide a graphical interface to define operations on messages.

A message flow can be embedded in another flow. Embedded flows are known as *subflows*. Subflows allow common processing (for example, error handling) to be reused. A subflow appears as a single node in the main message flow. The Error_Handling node in Figure 6-8 on page ccxcviii represents the subflow shown in Figure 6-10.

Subflows are comprised of the same nodes as main flows, except that they have Input and Output Terminal nodes to connect them to the main flow.

![Figure 6-10 An error handling subflow](image)

**Wiring nodes together**

Each node has input and output terminals. Terminals vary with the node type, but typically you might have an “in” terminal that receives the message from the previous node or transport, an “out” terminal used to pass the message along to the next node for processing, and a “failure” terminal that handles failure conditions.

Nodes are strung together to form a message flow by wiring an output terminal of one node with the input terminal of another; see Figure 6-11 on page ccci.
Message node types

WebSphere Business Integration Message Broker comes with a set of built-in nodes ready to use in building message flows. The built-in nodes shown in Table 6-1 show the broad range of function provided.

Table 6-1 Message manipulation nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes for message manipulation</td>
<td></td>
</tr>
<tr>
<td>▶️ Compute</td>
<td>Changes a message or contents of a database using ESQL.</td>
</tr>
<tr>
<td>▶️ Database</td>
<td>Used to interact with an ODBC datasource.</td>
</tr>
<tr>
<td>▶️ DataDelete</td>
<td></td>
</tr>
<tr>
<td>▶️ DataInsert</td>
<td></td>
</tr>
<tr>
<td>▶️ Dataupdate</td>
<td></td>
</tr>
<tr>
<td>▶️ Warehouse</td>
<td></td>
</tr>
<tr>
<td>▶️ Extract</td>
<td>Used to extract the exact contents of the input message that you want to be processed by later nodes in the message flow.</td>
</tr>
</tbody>
</table>
If you need to perform processing that is not supported by the built-in nodes, WebSphere Business Integration Message Broker allows the development of custom user-defined nodes. These can be written in either C or Java.

Additional nodes are available as SupportPacs and are free to download from the IBM Web site.

<table>
<thead>
<tr>
<th>Node</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>► Mapping</td>
<td>Use the Mapping node to construct one or more new messages by creating new messages and populating them with new information, with modified information from the input message, or with information taken from a database.</td>
</tr>
<tr>
<td>► XMLTransformation</td>
<td>Applies a stylesheet to an XML message.</td>
</tr>
<tr>
<td>► AggregateControl</td>
<td>Used to combine the generation and fan-out of a number of related requests with the fan-in of the corresponding replies, and compile those replies into a single aggregated reply message.</td>
</tr>
<tr>
<td>▶ AggregateReply</td>
<td></td>
</tr>
<tr>
<td>▶ AggregateRequest</td>
<td></td>
</tr>
<tr>
<td>Nodes for decision making</td>
<td></td>
</tr>
<tr>
<td>► Check</td>
<td>Validates the format of a message.</td>
</tr>
<tr>
<td>► Filter</td>
<td>Routes a message based on conditional logic.</td>
</tr>
<tr>
<td>► FlowOrder</td>
<td>Used to control the order in which a message is processed by a message flow.</td>
</tr>
<tr>
<td>► Label, RouteToLabel</td>
<td>Use the Label node in combination with a RouteToLabel node to dynamically determine the route a message takes through the message flow, based on its content.</td>
</tr>
<tr>
<td>► ResetContentDescriptor</td>
<td>Used to request that the message is reparsed by a different parser.</td>
</tr>
<tr>
<td>Nodes for error handling</td>
<td></td>
</tr>
<tr>
<td>► Throw</td>
<td>Used to throw an exception within a message flow.</td>
</tr>
<tr>
<td>► Trace</td>
<td>Used to generate trace records that can incorporate text, message content, and date and time information, to help you to monitor the behavior of the message flow.</td>
</tr>
<tr>
<td>► TryCatch</td>
<td>Used to provide a special handler for exception processing.</td>
</tr>
</tbody>
</table>
Note: Always check that a specific node is compatible with all of the target platforms on which the message flow will be deployed.

Transport support for applications
Table 6-2 summarizes the available transport mechanisms and their supporting nodes that allow clients to communicate with applications through message flows.

Table 6-2 Application transport support

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCADAInput, SCADAOutput</td>
<td>WebSphere MQ Telemetry Transport is a lightweight publish/subscribe protocol flowing over TCP/IP. This protocol is used by specialized applications on small footprint devices that require a low bandwidth communication, typically for remote data acquisition and process control.</td>
</tr>
<tr>
<td>Real-timeInput, Real-timeOptimizedFlow, Publication</td>
<td>WebSphere MQ Multicast Transport is used by dedicated multicast-enabled JMS application clients to connect to brokers. Applications communicate with the broker by writing data directly to TCP/IP ports. This protocol is optimized for high volume, one-to-many publish/subscribe topologies.</td>
</tr>
<tr>
<td>HTTPInput, HTTPReply, HTTPRequest, Publication</td>
<td>WebSphere MQ Web Services Transport allows Web services clients using XML messages and the HTTP protocol running over TCP/IP to communicate with applications through message flows in a broker.</td>
</tr>
<tr>
<td>Real-timeInput, Real-timeOptimizedFlow, Publication</td>
<td>WebSphere MQ Real-time Transport is a lightweight protocol optimized for use with non-persistent messaging. JMS applications communicate with the broker using TCP/IP ports.</td>
</tr>
<tr>
<td>MQInput, MQOutput, MQReply, Publication</td>
<td>WebSphere MQ Enterprise Transport supports WebSphere MQ applications that connect to WebSphere Business Integration Message Broker by writing data to and reading data from message queues.</td>
</tr>
</tbody>
</table>
6.5.2 Publish/subscribe applications

A robust message delivery system like WebSphere MQ can disseminate information to multiple users who may have an interest in some or all of that information. This function is provided by WebSphere MQ and the WebSphere Business Integration Brokers using the Publish/Subscribe programming model, relieving a message-producing application from managing multiple point-to-point connections. Program complexity is reduced since the model decouples a message producer from a set of message consumers.

The Publish/Subscribe paradigm involves one or more publishers who produce messages that are exchanged with a broker, and a group of subscribers who subscribe to some or all of the published messages managed by the broker. The broker matches the publications to the subscribers based on a topic and makes sure that all relevant messages are made available to the appropriate subscribers in a timely manner. The concept is illustrated in Figure 6-12 on page cccv.

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Transport</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQeInput node</td>
<td>WebSphere MQ Mobile Transport is used exclusively by WebSphere MQ Everyplace clients. WebSphere MQ Everyplace is an application designed primarily for messaging to, from, and between pervasive devices. These are typically small, handheld devices, such as mobile phones and PDAs. A bridge queue on the broker's queue manager provides an interface for the WebSphere MQ Everyplace clients to the broker services.</td>
</tr>
<tr>
<td>MQeOutput node</td>
<td></td>
</tr>
<tr>
<td>Publication</td>
<td></td>
</tr>
</tbody>
</table>
As shown in Figure 6-13 on page cccvi, brokers can be connected together, enabling efficient message exchange. In reality, one broker becomes a subscriber to another. Acting as a surrogate, each broker represents its respective subscribers, requiring a subscriber to interact with only one broker. Furthermore, network traffic is reduced as only one copy of a published message is exchanged between any two connected brokers.

Subscribers are able to choose among published messages based on the topic. With the WebSphere Business Integration Brokers, message content filtering is also provided.

Topic designation is often expressed as a namespace, permitting subscribers an opportunity to fine-tune subscriptions based on a qualified topic. The WebSphere Business Integration Brokers further offer the use of wild cards in both topic and content filter specifications.

Topic designation appears in an MQRFH or MQRFH2 header (referred to here as an RFH header). Header selection depends on the level of broker being used. The RFH header also supports a command syntax permitting publishers to register, publish or deregister a topic. The WebSphere Business Integration
Brokers offer further options, such as indicating whether a message is to be delivered immediately or retained by the broker.

![Extended Publish/Subscribe Example](image)

Figure 6-13   Extended publish/subscribe example

Similarly, subscribers can register or deregister a subscription, with or without a content filter, on a topic. Subject to the local security policy, there are no restrictions on the number of topics publishers and subscribers may deal with.

Various WebSphere MQ functions are exploited by the publish/subscribe model to include assured message delivery through the use of persistent messages, implicit temporary subscriptions when associated with a temporary dynamic queue, and message expiration to ensure data timeliness.

Programming interfaces, implicit and explicit, are supported through the WebSphere MQ Message Queue Interface (MQI), the WebSphere MQ Application Messaging Interface (AMI) available as SupportPac MA0F, and the Java Message Service (JMS) included in the Java 2 Enterprise Edition (J2EE) specification defined by Sun Microsystems. The current version of WebSphere MQ (V 5.3) is delivered with Java support, whereas earlier versions require the
addition of the MQSeries Java Client SupportPac, MA88. SupportPacs can be downloaded from:

http://www.ibm.com/software/integration/support/supportpacs/

Complex broker networks can be created to improve delivery efficiency.

And in spite of differing functionality, WebSphere MQ pub/sub brokers and WebSphere Business Integration Brokers can directly exchange published documents. WebSphere Business Integration Brokers can further be organized into *collectives*, where a number of broker queue managers are fully interconnected, thereby optimizing the communications between brokers (via their respective queue managers).

Explicit pub/sub support is the foundation for the WebSphere Business Integration Event Broker. While the pub/sub function is shared with the WebSphere Business Integration Message Broker, Event Broker is an alternative when the requirement is *only* for pub/sub application support and not message brokering in general.

The WebSphere Business Integration Event Broker can perform as a streamlined protocol converter, accepting input and producing output in any of the following forms:

- WebSphere MQ messages
- WebSphere MQ Mobile Transport messages (from WebSphere MQ Everyplace applications)
- Telemetry messages from SCADA (Supervisory, Control And Data Acquisition) devices
- MQ Real-Time Transport or MQ Multicast Transport messages produced by JMS applications

To prevent misuse of the publish aspect of pub/sub, system administrators are encouraged to implement access control functions on those queues from which the broker gets the messages to be published. Failing to do so could lead to a flood of unwanted messages, severely impacting network capabilities.

Depending on the platform, WebSphere MQ interfaces with operating system security functions, such as RACF on z/OS or the WebSphere MQ Object Authority Manager available on most distributed platforms, when queue access is requested by a user. Access control will permit queue usage by an authorized user, while denying use to an unauthorized user.

The WebSphere Business Integration Brokers provide an added level of security in the form of access control at the topic level. In conjunction with access control at the queue level, topic level security permits users, based on user ID or security
group membership, to access the publication queue but only to produce messages on designated topics. When this level of security is required, a User Name Server must be configured into the WebSphere Business Integration Message Broker environment.

Additional information on publish/subscribe in a message broker environment can be found in the IBM Redbook, *MQSeries Publish/Subscribe Applications*, SG24-6282.

### 6.6 Message Brokers Toolkit

In the previous topics you have already caught a glimpse of the Message Brokers Toolkit and how it fits into the equation. Now let’s take a look at the Toolkit itself.

The Message Brokers Toolkit consists of a Workbench window which displays one or more perspectives. A *perspective* is a group of views and editors required to perform tasks associated with a role. Each perspective consists of views that provide alternative presentations of resources or ways of navigating through information in the Workbench. As a user works with the Workbench, the data representing the projects and working environment are stored in a workspace directory on the local file system.

#### 6.6.1 Message flow development

The Message Brokers Toolkit provides the tools required to develop, deploy, and test message flows. Message flow development is done using the Broker Application Development perspective (shown in Figure 6-14 on page cccix).
Figure 6-14   Broker Application Development perspective

This perspective is used to develop and test message flows and message sets. You can see in Figure 6-14 that the perspective consists of views designed specifically for development. The Resource Navigator view (top left) lists the message flow resources. The palette (top right) is the working area where nodes are placed in the flow, node properties are set, and the nodes connected together. The Tasks view (bottom right) shows informational, warning, and error messages that indicate problems within resources. The Outline view displays a structured outline of the file open in the editor area (palette).

6.6.2 Broker administration

The Message Brokers Toolkit provides broker administration functions through the Broker Administration perspective (Figure 6-15 on page cccx).
6.6.3 Deploying message flows

At runtime, a message flow is deployed to an execution group in a message broker. Multiple copies of a message flow can run at the same time to process more messages, either by assigning the same flow to multiple execution groups, or by increasing the number of instances of the flow which run in a single execution group.
The unit of deployment is the broker archive file (BAR) file. Before deploying, you must create a BAR file and add your message flows to it (Figure 6-16). You then deploy the BAR file to the execution group.

![Figure 6-16 Working with BAR files](image)

Messages will appear indicating the success or failure of the deploy request to the Configuration Manager, and the Event Log will contain records indicating the success or failure of the deploy to the execution group (Figure 6-17 on page cccxii).
When you make future changes to the message flows, you simply update the BAR file and redeploy.

Note: For rapid testing, there is a feature called rapid application development (RAD). RAD enables you to configure a server project so that you can repeatedly deploy a server configuration (containing your message flows and message sets) to the target broker execution group with a single toolbar button click. When you deploy, a temporary BAR file is created and deployed to the broker.

6.6.4 Debug facilities

There are two primary methods of debugging message flows. The Tasks view is dynamically updated as message flows are developed, indicating any problems found during validation by the editors. The second method is a runtime debugging facility using breakpoints.
Tasks view
The Tasks view provided in the Broker Application Development perspective lists outstanding problems in the currently open projects. These tasks are created as you develop message flows by the automatic validation processes. Double-clicking an entry in the Tasks view will open the file where the problem is and set the current pointer to the problem entry. In addition, you can create custom tasks as a reminder to carry out actions at a later date.

Message flow debugger
The Message Brokers Toolkit provides a message flow debugging utility that allows the developer to set breakpoints between the nodes in a message flow and after the output nodes, allowing the contents of the message to be examined and altered as it passes through the flow. It also includes the ability to set breakpoints and step through the ESQL used by the nodes in a flow.

Flow Debug perspective is used to set breakpoints and execute the flow in debug mode (Figure 6-18).

Figure 6-18  Breakpoint encountered
6.6.5 Team development

CVS is an open source team resource management system that uses a branch model to support multiple courses of work that are somewhat isolated from each other but still highly interdependent. This model allows individuals to work on a team project, share their work with others as changes are made, and access the work of others as the project evolves.

Users can commit their work to the CVS repository, synchronize or catch up their local resources with resources in the repository, compare and merge local and repository resources, and exploit many more advanced functions.

The Message Brokers Toolkit provides the interface to a CVS repository through the CVS Repository Exploring perspective.

6.7 For more information

For more detailed information on using WebSphere Business Integration Message Broker and its capabilities, we recommend the following IBM Redbooks:

- *Migration to WebSphere Business Integration Message Broker V5*, SG24-6995:
  
  http://www.redbooks.ibm.com/abstracts/sg246995.html

- *Using Web Services for Business Integration*, SG24-6583:
  
  http://www.redbooks.ibm.com/abstracts/sg246583.html

- *Patterns: Broker Interactions for Intra- and Inter-enterprise*, SG24-6075:
  
  http://www.redbooks.ibm.com/abstracts/sg246075.html
WebSphere Studio V5
overview and architecture

WebSphere Studio provides a comprehensive development environment designed to meet all of your development needs—from state-of-the-art Web interfaces to server-side applications, from individual development to advanced team environments, from Java development to application integration. Available in a number of configurations, with extensions from IBM and partners, WebSphere Studio enables developers to use a single development environment designed to meet their specific development needs.

From one development environment you can develop, test, deploy, and manage applications. WebSphere Studio is designed to be used in wide variety of development roles such as Web developer, Java developer, enterprise programmer, business analyst, and system architect. A rich set of utilities and wizards helps simplify common tasks so that developers can concentrate on providing true business value and on rapidly getting robust applications into production.

This discussion centers around Version 5.1.2 of WebSphere Studio Site Developer and WebSphere Studio Application Developer, and Version 5.1 of WebSphere Studio Application Developer Integration Edition and WebSphere Studio Enterprise Developer.
7.1 WebSphere Studio family overview

WebSphere Studio configurations are all built on the WebSphere Studio Workbench that extends the open-source Eclipse platform and provides an open, extensible plug-in architecture. Numerous plug-ins are available from partners and the open source community, or, using the included plug-in development environment, you can create your own plug-ins for specific needs.

WebSphere Studio is currently available in four configurations. Each configuration contains a set of IBM-provided plug-ins that are suited to specific types of application development:

- Site Developer contains a set of plug-ins that provide the Web developer with tools for developing dynamic Web applications.
- Application Developer includes all Site Developer plug-ins plus a richer set of plug-ins and more advanced features for the Java 2 Enterprise Edition (J2EE) developer.
- Application Developer Integration Edition adds features for access to Enterprise Information System (EIS) resources through the Java Connector Architecture (JCA).
- Enterprise Developer targets users requiring mainframe development tools and applications requiring z/OS runtime support.

7.1.1 Eclipse platform

Eclipse is an open-source platform that has been designed from the ground up for building integrated Web and application development tools. By design, the platform does not provide a great deal of end-user functionality by itself. The value of the platform is that it encourages rapid development of integrated features based on a plug-in model.

A tool for building tools that build applications

You can think of the Eclipse platform as an empty, well-designed workbench that helps tool builders create tools and organize them so that other users can more easily concentrate on doing their jobs of actually building end-user widgets.

The well-designed workbench has numerous advantages. It has standard-sized slots and connectors that are ready to accept all sorts of tools. The tools can connect and communicate with each other. The workbench looks and acts the same in multiple environments. Tools can share resources.

The Eclipse platform uses the common workbench model to integrate the tools from the user's point of view. A workbench user becomes accustomed to using
the same user interface for all tools in the workbench even if they come from different tool suppliers.

**Plug-in architecture**

At the core of Eclipse is a plug-in architecture model that permits extensibility and dynamic discovery of plug-ins. Tool developers can write plug-ins that interface with the workbench using well-defined hooks called extension points. The extension model allows plug-in developers to add support for additional file types and customized editors. The artifacts for each tool, such as files and other data, are coordinated by a common platform resource model.

**Multiple operating systems**

Eclipse is designed to run on multiple operating systems (Windows, Linux) while providing robust integration with each underlying operating system. Plug-ins use the Eclipse portable APIs and run unchanged on any of the supported operating systems.

### 7.1.2 Eclipse architecture

Figure 7-1 depicts the Eclipse architecture. It shows the plug-in nature of the environment and how tools can connect to the platform through extension points.

![Eclipse architecture diagram](image)

*Figure 7-1  Eclipse architecture*
The platform is structured as subsystems built on a small runtime engine. Subsystems are implemented by one or more plug-ins. Also shown is the plug-in development environment (PDE) that allows tool builders to develop plug-ins using the Java development tools.

Here is a short description of subsystems:

- **Platform runtime**
  
  Defines the extension point and plug-in model. It dynamically discovers plug-ins and maintains information in a platform registry. Plug-ins are started up when required according to user operation of the platform.

- **Workbench**
  
  Implements the user interface for navigating the resources and using the tool plug-ins. It defines extension points for adding user interface components such as views or menu actions. It includes JFace and Standard Widget Toolkit (SWT) for building user interfaces.

- **Workspace (resource management)**
  
  Defines an API for creating and managing resources (projects, files, and folders) that are produced by tools and kept in the file system.

- **Team support**
  
  Defines a team programming model for managing and versioning resources.

- **Help system**
  
  Defines extension points for plug-ins to provide help or other documentation such as browsable books.

- **Debug support**
  
  Defines a language-independent debug model and UI classes for building debuggers and launchers.

**Base workbench features**

The Eclipse platform has common features that are present in all WebSphere Studio family products.

**Open standards**

The whole Eclipse workbench, as well as all products of the WebSphere Studio family of products, are built on open standards, and the code that they generate also complies with open standards. This allows you to build and deploy state-of-the-art, server-side applications that conform to the Servlet 2.3, JSP 1.2, and Enterprise JavaBeans 2.0 specifications.
**Vertical and horizontal integration**
Traditionally software vendors have provided vertical tools, forcing customers to do their own integration. Eclipse provides a platform that software vendors can easily extend. Vendors have embraced this technology and are continuing to actively build tools on this foundation.

The WebSphere Studio product that is built on Eclipse offers tools that are already seamlessly integrated, freeing you to focus on building applications rather than interfacing or bridging between tools used by different roles (business analyst, graphic artist, Java programmer, tester, COBOL programmer).

**State-of-the-art integrated development environment**
WebSphere Studio has won many awards as an integrated development environment for Java, Web services, and XML as the number of experienced users of WebSphere Studio continues to increase.

**Multiple operating systems**
WebSphere Studio runs on multiple operating systems:
- Linux (RedHat V7.2 and V8.0, Suse V7.2 and V8.1)
- Windows NT®, 2000, XP

**Role-oriented development**
The Workbench is designed to support particular e-business development roles. Roles are supported by task-oriented perspectives, which combine views and editors of project resources. Perspectives filter out complexity, and present the developer with only those functions that are relevant to the task at hand.

A Web page designer, for example, has easy access to the tools required for that job in the Web perspective, and is not forced to search for Web development tools in a cluttered user interface that presents a myriad of other tools that are not used in the Web developer role. A Java developer, in contrast, works most often in the Java perspective.

**Programming performance and productivity**
A common look and feel across project roles and perspectives reduces the learning curve and increases user productivity.

Team productivity and application quality improve through greater automation of the development and deployment process via an integrated development environment. Many productivity enhancing features are included in all configurations, such as:
- A tool to visually manage a Web site
- Wizards to generate artifacts that comply with the servlet and J2EE specifications
- Automatic updates to Web links as you move resources around
- Rich media tools that enhance images and create JavaScript to add visual interest to Web sites
- Built-in refactoring tools that make it easy to reorganize Java classes
- An integrated unit test environment with hot method replace for WebSphere Application Server and Apache Tomcat
- Profiling tools that help optimize applications by detecting and analyzing performance problems

**Multiple software configuration management repositories**
WebSphere Studio supports multiple Software Configuration Management (SCM) repositories. Several SCM team adapters are available from the open source community, IBM, and other commercial vendors.

**Support for multiple target runtime environments**
Developers can develop applications for multiple target runtime environments on multiple platforms. Java Development Kits (JDKs) from IBM and other vendors can be used in development.

**Open team development**
The team development environment for all products based on the Eclipse Workbench supports pluggable repositories rather than mandating any proprietary repository. An optimistic concurrency model is supported. All development resources for all projects can be stored in a single repository; therefore, developers have consistent team support for their projects, and are able to easily share their work products.

### 7.1.3 WebSphere Studio Workbench

WebSphere Studio Workbench is IBM’s implementation of Eclipse. It enables tool providers to build platform-neutral, professional, enterprise-ready application development tools that interoperate seamlessly with the other Workbench tools.

The WebSphere Studio Workbench provides a tool development kit so that tool builders can create plug-ins that target the WebSphere platform. WebSphere Studio configurations plus other vendor plug-ins are easily combined in one environment. The result is an integrated application development environment that provides a consistent look and feel in a package tailored for an enterprise’s needs.
The following sections describe WebSphere Studio Workbench features and functions that are common to all the configurations in the WebSphere Studio family.

Figure 7-2 shows a typical WebSphere Studio Workbench implementation.

![WebSphere Studio Workbench](image)

**Perspectives**

Several perspectives are provided with each configuration of WebSphere Studio. A perspective defines an initial set and layout of views and editors for performing particular development activities. You can change the layout of views, change preferences, and save a perspective that you have customized, so that you can open it again later. A customized perspective can be shared with other members of a project.

You can open more than one perspective at a time, and switch perspectives with one click as you are working.

**Views**

Views provide alternative presentations of resources or ways of navigating through information in the workbench. For example, the Navigator view displays projects and other resources that you are working with in a folder hierarchy. A
view might appear by itself, or stacked with other views in a tabbed notebook arrangement.

A perspective determines the views that you are likely to need, but you can also add views to a perspective. For example, the Java perspective includes the Packages view and the Hierarchy view to help you work with Java packages and hierarchies. You may want to add the Servers view to start and stop an application server.

Editors
When you open a file, WebSphere Studio automatically opens the editor that is associated with that file type. For example, an HTML editor is opened for .html, .htm, and .jsp files; a Java editor is opened for .java files; and a text editor is opened for .txt files. Editors that have been associated with specific file types open in the editor area of the workbench.

By default, editors are stacked in a notebook arrangement inside the editor area. You also have the option of tiling open files. However, if there is no associated editor for a resource, WebSphere Studio will attempt to launch an external editor outside the workbench. If you want to open a resource with a different editor, select Open With from the context menu.

Projects
A project is the top level of organization of resources in the workbench. A project contains files and folders that are grouped into buildable units. In the workbench, all folders and files must be contained in projects. Projects are used for building, version management, sharing, testing, and deployment.

You can create several different types of projects in WebSphere Studio. Web, Java, and Enterprise application projects are some examples. Different types of projects have different structures, different associated builders, and different automatic validation routines.

When you create a project, you give it a file system location. By using an operating system tool to look at this location outside the workbench, you can view the project as a file system directory, and its contents as files or sub-directories.

A project's build specifications control how the project is built, including the steps required to convert the project resources from their authoring form to their execution form. A project can contain session and persistent properties, settings for environmental variables, and references to other projects.

Builders create or modify workspace resources within projects, usually based on the existence and state of other resources. They are a mechanism for enforcing
the constraints of some domain. For example, a Java builder compiles Java source files into executable class files, and a Web link builder updates links to files when names and locations have changed. As resources are created and modified within a project or projects, builders can be set to automatically run, and the constraints are maintained.

**Team development**

The team development environment in WebSphere Studio supports file-based pluggable repositories instead of forcing users to use only one dedicated proprietary repository. A file-based pluggable repository allows teams to easily integrate favorite tools and provides flexible source management.

The source configuration management systems that are packaged with WebSphere Studio are Concurrent Versions System (CVS) and Rational ClearCase LT (not in the Site Developer configuration). With WebSphere Studio Enterprise Developer there is also a plug-in for Software Configuration Library Manager (SCLM).

**Other software configuration management repositories**

Other SCM repositories can be integrated with WebSphere Studio as extensions of the Eclipse SCM team adapter. SCM team adapters for commercial SCM products are provided by the vendors of those products. To find a list of SCM products and adapters provided by IBM Business Partners, go to the SCM team adapters page of the product site. IBM does not endorse or warrant these adapters. For support, or further information, please contact the SCM product vendor directly.

**Sharing resources**

In WebSphere Studio, developers do all of their work in their individual workspaces. When a developer is ready to share their work with others they release their changes to their underlying SCM. This model allows individual developers to work on a team project, share their work with others as changes are made, and access the work of other developers as the project evolves.

**Object ownership**

Unlike some proprietary repository systems, WebSphere Studio does not enforce any notion of object ownership. Write access is controlled by whether you have write permission to the directory that holds the resource. Anyone who is authorized to the resource can create, edit, or delete the resource. Typically each SCM user has an account on the server so that actions are logged to particular user IDs. The creation of users and groups is done by the SCM server administrator.
**Source location and local history**

Every time you save a resource, the changes are made to the workbench's file system, which will usually be a local file system. By default the files are stored in a directory called workspace. You can maintain different sets of resources in different workspaces on a file system, and specify which workspace you wish to work with when you start WebSphere Studio.

Even without the support of an external SCM, a history of the files that have changed can be cached locally, so that the resource can be compared to and replaced with earlier versions of itself. These local histories are independent of the team development environment and the repository. You can set the amount of disk space that is used for local history information in the preferences.

**The online help system**

Most of the documentation for WebSphere Studio is online. Online help is organized to help you find the right information when and where you need it. The help browser opens in a separate window, allowing you to use the workbench and view the help at the same time. A search facility is available to find a word or phrase in the documentation. You can choose from different sets of information; some are intended for application developers and others are intended for ISVs and other developers of plug-in tools.

Tool developers can easily add to the existing help system because items in the Eclipse table of contents are in fact plug-ins that use help system extension points.

### 7.1.4 WebSphere Studio Site Developer

Site Developer meets the needs of content authors, graphic artists, Web programmers, and Web masters. This integrated tool makes it easy to collaboratively design, create, assemble, publish, deploy, and maintain dynamic, interactive Web applications. You can quickly build and test business logic, and enhance presentation artifacts with built-in Web creation tools before deploying on a production server such as WebSphere Application Server or Apache Tomcat.

**Note:** Site Developer is also included in WebSphere Application Server - Express, but a few features have been modified or eliminated to tailor it for the Express environment. These will be noted at the appropriate places.

This configuration includes an intuitive WYSIWYG editor that lets novice Web designers create and publish Web sites while incorporating the latest Web technology, including JavaScript, dynamic HTML, and Cascading Style Sheets.
Visual layout tools are provided to create dynamic Web sites with servlet or JSP components. Included is a built-in XML development environment along with Web services tools for the creation, deployment, and publishing of Web services.

Use of Site Developer encourages the Model-View-Controller (MVC) approach to architecting a Web application:

- Developers create the Model, which is composed of the business logic that typically interacts with corporate data stores.
- Web designers and graphic artists create the View, which is composed of user-interface components such as JSP pages, HTML files, and GIF files.
- Application architects create the Controller, which is composed of the software that oversees the flow of runtime events at the highest level.

You can develop Web applications that employ the following technologies:

- **JavaServer Pages (JSP)**
  A simple, fast, and consistent way to extend Web server functionality and create dynamic Web content. A WYSIWYG editor of JSP pages enables rapid development of Web applications that are server- and platform-independent.

- **Servlets**
  Server applications that execute within a Web application. Site Developer supports the Java Servlet 2.3 specification.

- **Web services**
  Self-contained, modular applications that can be described, published, located, and invoked over the Internet or within intranets.

- **Struts**
  A set of Java classes and JSP tag libraries that provide a conceptual framework for developing Web applications.

- **JavaServer Faces**
  A set of APIs, UI components, and JSP tag libraries that provide a framework for managing UI state across server requests and offering a simple model for the development of server-side events that are activated by the client.

- **V5.1.2 of Site Developer and Application Developer also now include EGL tools that generate Java code.**

Instead of using several different products for graphics, JSP files, business logic, testing, and deployment, the entire Web development team can use Site Developer to design, create, test, and deploy advanced, end-to-end Web applications.
7.1.5 WebSphere Studio Application Developer

Application Developer is designed for professional developers of Java applications conforming to Java 2 Platform, Enterprise Edition (J2EE) specifications, who require integrated Java, Web, XML, and Web services support. It includes all of the features of Site Developer, and adds tools for developing Enterprise JavaBean (EJB) applications, as well as performance profiling and logging tools for both local and remote execution. The target runtime environment is a J2EE server such as the WebSphere Application Server Network Deployment configuration to take full advantage of EJB capabilities.

Using the performance profiling and tracing tools, it is possible to detect application performance bottlenecks early in the development cycle. The built-in test environment for WebSphere Application Server and advanced tools for code generation help shorten the test cycle.

7.1.6 WebSphere Studio Application Developer Integration Edition

Application Developer Integration Edition builds on the features of Application Developer, and adds the following tools for integration with back-end systems:

- Enterprise services, a set of tools and wizards to facilitate service-oriented development that provides a consistent view of any service regardless of implementation. Enterprise services provide the tools and support for consuming services from and providing services to WebSphere Business Integration Server Foundation, which acts as the point of integration for a wide variety of services.

  Enterprise services allow you to consume various resources, such as Simple Access Object Protocol (SOAP) Web services, Java beans, stateless session (EJB) beans, and J2EE Connector Architecture (JCA) services.

  Services deployed into the WebSphere Application Server can be provided as SOAP services via the EJB programming model. Enterprise services are based on open standards such as J2EE, JCA, Web Services Description Language (WSDL), and Extensible Stylesheet Language Transforms (XSLT).

- Graphical business process composition tools that enable you to visually compose a service out of one or more existing services, define and transform the flow of information between services, and create a business process that contains other nested processes. Support for Business Process Execution Language for Web Services (BPEL4WS) and Flow Definition Markup Language (FDML) is included.

- For integration with back-end systems, Application Developer Integration Edition provides development connectors for Customer Information Control System (CICS), Information Management System (IMS), and
Host-on-Demand (HOD). IBM provides a range of adapters for popular packaged applications including Customer Relationship Management, Supply Chain Management, Enterprise Resource Planning, and mainframe applications.

- Application Developer Integration Edition also provides support for the full set of Enterprise services provided by WebSphere Application Server Enterprise. For example, internationalization support allows you to build a single application that automatically adjusts currencies, date formats, language, etc.

### 7.1.7 WebSphere Studio Enterprise Developer

Enterprise Developer provides enterprise-level development tools for interacting with back-end development and operating systems. In addition to Application Developer Integration Edition features, Enterprise Developer provides the following:

- z/OS application development tools in an interactive, workstation-based environment where you can do the following tasks:
  - Perform MVS-based system tasks such as managing datasets with a workstation-like directory structure and submitting JCL or TSO commands.
  - Create HLAASM, COBOL, or PL/I code, including CICS, IMS, and SQL statements.
  - Edit, test, and compile the source code locally or remotely on a z/OS system.
  - Transfer CLISTs and REXX EXECs scripts to z/OS, run them, and view the resultant output.
- Integration with existing transactional environments such as CICS and IMS.
- Tools for creating new Enterprise Information System (EIS) applications as part of a J2EE application.
- The ability to integrate Struts-based Model-View-Controller applications using J2EE Connector Architecture connectors and EGL.
- Integration with WebSphere Studio Asset Analyzer (WSAA) to scan and identify z/OS or OS/390 artifacts and distributed assets that make up core business application processes and connecting points. Data is stored in a DB2 database along with relationships and links among artifacts. Database information can be accessed through a Web browser to help find the affected source code and assess the scope of a change.

WSAA provides impact analysis of assets to quickly determine the ramifications of changes before they are implemented. WSAA also provides
the ability to search, find, and extract an existing segment of COBOL code and then generate new components for reuse.

**Note:** Previous releases of WebSphere Studio Enterprise Developer included EGL support for the creation of full-function Java and COBOL applications. This support has been removed in WSED 5.1.1. IBM plans to again add EGL Language Support, with significant extensions, to WebSphere Studio Enterprise Developer later in 2004. These extensions will provide additional new application development capabilities for both Web and traditional text applications as well as enable migration and execution of existing VisualAge Generator V4.5 applications for selected platforms and data sources.

### 7.1.8 Functional comparison: Technology support

Table 7-1 shows a comparison of the various WebSphere Studio configurations and their features.

**Table 7-1  WebSphere Studio functional comparison**

<table>
<thead>
<tr>
<th>Feature</th>
<th>WSSD 5.1.2</th>
<th>WSAD 5.1.2</th>
<th>WSAD-IE 5.1</th>
<th>WSED 5.1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug-in development tools</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Web development tools (JSP and servlet)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Struts tools</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>JavaServer Faces tools</td>
<td>Yes</td>
<td>Yes</td>
<td>beta</td>
<td>beta</td>
</tr>
<tr>
<td>XML tools</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Relational database tools</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Java development tools</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Enterprise JavaBean tools</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Web services development tools</td>
<td>Yes (no EJB Web service support)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Team collaboration tools (includes CVS and Rational ClearCase LT)</td>
<td>Yes (CVS only)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Integrated debugger</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Profiling and logging tools</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
7.2 Plug-in development tools

The Plug-in Development Environment (PDE) is designed to help you develop platform plug-ins. PDE is installed in the workbench and includes a perspective, wizards, editors, and views such as the error log and plug-in registry that streamline the process of developing plug-ins inside the workbench. A PDE wizard for plug-in creation allows you to create a plug-in project using templates and invoke a code generation wizard that will give you plug-ins with various editors, a menu, a property page, or a view. An editor is provided to create the plug-in manifest file that contains information about the plug-in such as other plug-in prerequisites, runtime libraries needed, and extension points to extend the new plug-in.

Using the PDE, developers are guided in creating plug-ins that conform to the workbench’s clearly defined plug-in structure and specification. This assures that the common programming rules of the platform are followed and that a plug-in is correctly incorporated into the environment. For example, a plug-in project in the workspace requires certain files like the manifest file that identify it as a plug-in.

7.2.1 Host and runtime workbench instances

An important concept in plug-in development is the notion of host and runtime workbench instances. When you start up the workbench you are running the host workbench instance. You will use it to work on projects that define plug-ins you are building. You can launch the runtime instance of the workbench to test the plug-in.
7.2.2 Typical plug-in development scenario

The typical plug-in development scenario is to create a plug-in using the host workbench instance and the plug-in creation wizard; then, once you are happy with the plug-in and want to test it, you launch another workbench instance, the runtime instance. This instance will contain the same plug-ins as the host instance, but in addition will also have the plug-ins you were working on in the host instance. PDE launcher will take care of merging the plug-ins with the host plug-ins and creating the runtime instance.

7.3 Web development tools

The following tools are the primary means for developing Web applications in the workbench.

7.3.1 Wizards

WebSphere Studio offers many wizards that help a Web designer add static or dynamic content to Web pages. Using the basic wizards you can create and specify settings for Web projects, HTML files, and JSP files, and use the examples provided with the product. Advanced wizards help you develop Web applications that retrieve information from common databases, use Java bean APIs to query or update databases, and generate output forms and servlets.

The wizards provide graphical user interfaces to guide you through the choices for creating Web application components. For example, a wizard presents a sequence of steps to create SQL queries, choose Java methods and properties, and integrate them into the Web pages you are creating. You do not have to be an expert at SQL syntax or Java programming, because the wizards walk you through the process step-by-step, and then generate JSP and servlet code to include in the Web applications.

7.3.2 Web Site Designer

Web Site Designer allows you to quickly build complex Web sites. It has a Site Map editor (as shown in Figure 7-3 on page cccxxxi) that gives you a graphical view of all the pages in a Web site. The navigation support enables you to begin a site by designing its organization, and then populating it with the appropriate Web pages, adding, deleting, and rearranging pages as you go. Web Site Designer also enables easy Web site reorganization by simply dragging and dropping pages or navigation branches in the site map.
A Site Parts palette is available with common Web elements that can then be added to a page with a simple drag and drop.

7.3.3 Page Designer

Page Designer is an advanced-function HTML and JSP editor that allows you to quickly build complex Web pages, both visually and textually. In addition to being a WYSIWYG editor with all standard HTML tags, Page Designer provides dynamic element support that enables you to include form elements, Java applets, embedded scripts, and JSP tags. An HTML Syntax Validator is included.
to validate HTML syntax and HTML Document Type Definition (DTD) compliance.

When you open a page with Page Designer there are three different tabs available with views of the page: Design, Source, and Preview.

**Design**
The Design page is the WYSIWYG editor that enables you to create and work with a file while viewing its elements on the page. For example, you can see the graphics that you have inserted into the file and continually check the visual presentation of the Web page as you design it. The Design page provides full visual editing capabilities, including the ability to drag and drop from the J2EE Navigator view, the Thumbnails view, and Windows Explorer.

**Source**
The Source page enables you to view and work with a file's source code directly. You are able to see and modify HTML and JSP tags. The Outline and Properties views both have features that supplement the Source page.

**Preview (available only for Windows)**
Preview shows you how the current page is likely to look when viewed in an external Web browser. Dynamic content will not be presented since it is served by an application server.

**Free layout mode**
An easy way for new users to lay out a Web page is to use free layout mode. It gives you the ability to lay out an HTML or JSP page by dragging objects from the Palette view and dropping them into their appropriate places on the page. Free layout mode uses free layout tables and cells, which are usually hidden from end users, to help position objects on the page. Once you create and size the table, you can place objects within cells of the table.

More experienced users may choose to use Standard mode, which is similar to laying out a page by coding in HTML tags.

### 7.3.4 Web Deployment Descriptor editor

The Web deployment descriptor editor helps you to define the deployment information for a Web application. It is a graphical editor that displays the contents of the Web deployment descriptor file named web.xml.
With the editor you can:

- Set numerous Web application parameters related to servlets, filters, listeners, mime mappings, parameters, references, security, and other general deployment settings.
- Edit the source code page if you prefer to work directly with the XML source code.
- Configure WebSphere Application Server-specific bindings and extensions, as appropriate.

The Web deployment descriptor is automatically generated when a new Web project is created and updated as new application elements are created using wizards.

### 7.3.5 JavaScript editor

The JavaScript editor provides a Source pane and, for Windows, a Preview pane. It features content assist editing, macros, customizable keywords, and syntax validation for both specific JavaScript versions and specific browsers.

### 7.3.6 Cascading Style Sheet (CSS) Designer

The CSS Designer has a Source pane and, for Windows users, a Preview pane. Styles in a CSS file can be edited with the help of syntax highlighting, content assist, and a preview function. The modifications made in CSS Designer are immediately applied to the Design page of Page Designer if the HTML file has a link to the CSS file.

CSS designer supports the following World Wide Web Consortium (W3C) cascading style sheet standards:

- CSS1 (cascading style sheet level 1)
- CSS2 (cascading style sheet level 2)
- CSS Mobile Profile 1.0

In addition, the Wireless protocol Cascading Style Sheet (WCSS) 1.0 specification, a CSS for WAP 2.0, is also supported. CSS Designer (and Page Designer for embedded CSS files) recognizes WCSS, as well as CSS2, in editing functions.

### 7.3.7 Pervasive computing support

WebSphere Studio provides extensive support for pervasive devices that rely on wireless standards such as Wireless Application Protocol (WAP), Compact HyperText Markup Language (CHTML), and Wireless Markup Language (WML).
You can visually create pages for popular wireless standards using a visual map for creating applications for cell phones, hand-holds, and other pervasive devices.

Page Designer emulates the target device in Design and Preview pages. The device profile includes screen size and default styles (fonts and margins).

**Page Designer XHTML support**
Page Designer supports HTML 4.01 (along with some legacy tags and attributes). However, it also supports XHTML standards for PvC-enabled application development. The following standards are supported:

- XHTML 1.0 (Strict, Transitional, and Frameset)
- XHTML 1.1
- XHTML Basic 1.0
- XHTML Mobile Profile 1.0 (defined in WAP 2.0)

### 7.3.8 Database Web Pages wizard

The Database Web Pages wizard uses servlet and JSP technology to generate Web pages from SQL queries. The wizard guides you through the process of building an SQL query, establishing a database connection, and configuring look and layout for list and details results pages.

Generated components include a servlet controller, and JSPs that provide a list and details view with an input form. Custom tag libraries are used by the JSPs to retrieve SQL data. Optionally, a Java bean wrapper can also be generated to invoke database accesses. Database accesses use a custom Java bean and Java Database Connectivity (JDBC) technology.

### 7.3.9 Java Bean Web Pages wizard

The Java Bean Web Pages wizard uses servlet and JSP technology to generate Web pages from Java beans. The wizard asks you to select the Java bean and its methods to be executed by the servlet. You also can select an error page and a style sheet, among other options. You will have to define the page and field properties to be exposed to the end user both for the input form as well as for the result form.

In addition to a controller servlet and JSPs, the wizard generates Java bean wrappers that set and retrieve bean properties, execute bean methods, and display method results.
7.3.10 Template Application wizard

The Template Application wizard is a tool that generates operational, ready-for-deployment Web applications based on predefined application templates. No Java programming skills are required to use this tool.

Upon opening the Template Application wizard, you are prompted to select a template and fill in the template's properties, based on the application you are going to build. The wizard presents only those properties that are specified as 'adjustable' in the template you selected.

An application template is really the skeleton of an application. It includes the entire application and may be used as generated or may have certain parts that remain to be specified. The generated code includes all the necessary JSPs and Java beans, along with additional required code artifacts.

Additional templates are available for download from the Web.

7.4 Struts tools

Struts is an open source subproject of the Apache Software Foundation's Jakarta project. It provides a runtime framework consisting of a set of Java classes and JSP tag libraries. Struts applications are based on the Model-View-Controller design pattern for Web applications executing in a J2EE environment.

WebSphere Studio provides a set of tools that assist in the development of Struts-based Web applications:

- Web projects can be configured to include Struts support (runtime classes, tag libraries, and other Struts-related resources) either at project creation or by editing the properties.
- Wizards are available to create Struts form beans, actions, exceptions, and configuration files.
- A specialized editor is provided to create and modify the Struts configuration file.
- A visual assembly tool, the Web Diagram editor, allows a developer to assemble forms, actions, JSPs, and HTML pages. Web diagrams can be used for designing and creating Struts applications or for visualizing and working with existing Struts applications.
- Editing and validation support for Struts tags is included in Page Designer.
- The Portal Toolkit provides support for Struts portlet development that is similar to Web application development.
7.4.1 Web Diagram Editor

The Web Diagram Editor allows you to visually construct a Struts-based Web application. The diagram is a graphical representation of the Struts configuration file. The editor has icons depicting the following:

- Struts module
- Web application
- Web page (HTML or JSP)
- Struts action mapping
- Struts form bean
- Java bean

You can add components to the diagram that do not exist. To create the component, simply double-click the component icon to open a “create” wizard for it. If the component already exists, you can open it for editing by double-clicking its icon. You can draw connections between the components to define navigation paths in the application.

7.4.2 Struts Configuration File Editor

The Struts Configuration File Editor allows a developer to declare components of Struts-based Web applications. The Struts configuration file is an XML file that defines the form beans, action mappings, message resources, and plug-ins like the Validator plug-in that are used in the Web application. The editor has several views that provide a user-friendly interface for defining the elements of the configuration file. There is also a Source view available for direct editing of XML source code.

7.5 JavaServer Faces tools

JavaServer Faces is a technology developed by Sun Microsystems that helps you build user interfaces for dynamic Web applications that run on the server. The JavaServer Faces framework manages UI state across server requests and offers a simple model for the development of server-side events that are activated by the client.

7.6 XML tools

WebSphere Studio has an XML perspective that gives developers access to many XML tools. The tool set includes editors and wizards that facilitate the use of XML application development technologies.
7.6.1 XML Editor

The XML Editor is a tool for creating and viewing XML files. You can use it to create new XML files, either from scratch, existing DTDs, or existing XML schemas. You can also use it to edit XML files, associate them with DTDs or schemas, and validate them.

7.6.2 Document Type Definition (DTD) Editor

The DTD Editor is a tool for creating and viewing DTDs. Using the DTD Editor, you can create DTDs, generate XML schema files, and generate Java beans. You can also use the DTD Editor to generate a default HTML form based on the DTDs you create.

7.6.3 XML Schema Editor

The XML Schema Editor is a tool for creating, viewing, and validating XML schemas. You can use the XML Schema Editor to perform tasks such as creating XML schema components, importing and viewing XML schemas, generating DTDs and relational table definitions from XML schemas, and generating Java beans for creating XML instances of an XML schema.

7.6.4 Extensible Style Sheet Language (XSL) Editor

The XSL Editor can be used to create new XSL files or to edit existing ones. You can use content assist and various wizards to help you create or edit the XSL file. Once you have finished editing the file, you can also validate it. You can associate an XML instance file with the XSL source file you are editing to help in defining constructions such as an XPath expression.

7.6.5 XPath Expression wizard

The XPath Expression wizard is used to create XPath expressions. XPath expressions can be used to search through XML documents, extracting information from the nodes (such as an element or attribute).

7.6.6 XSL debugging and transformation

The XSL debugging and transformation tool is used to apply XSL files to XML files, transforming them into new XML, HTML, or text files. After the transformation has taken place, the XSL Debug perspective opens, allowing you to visually step through an XSL transformation script, highlighting the transformation rules as they are fired. You can use the views in the XSL Debug perspective to help you debug the XML or XSL files.
7.6.7 XML and SQL query

You can use the XML and SQL Query wizard to create an XML file from the results of an SQL query or to take an XML file and store it in a relational table. When creating an XML file from an SQL query, you can optionally choose to create an XML schema or DTD file that describes the structure that the XML file has for use in other applications. Two Java class libraries, SQLToXML and XMLToSQL, are included for use in applications at runtime.

7.6.8 XML to XML Mapping Editor

The XML to XML Mapping Editor is a tool used to map one or more source XML files to a single target XML file. You can add XPath expressions, groupings, Java methods, or conversion functions to a mapping. Mappings can also be edited, deleted, or persisted for later use. After defining the mappings, you can generate an XSLT script. The generated script can then be used to combine and transform any XML files that conform to the source DTDs.

7.6.9 Relational database to XML mapping

The RDB to XML Mapping Editor is a tool for defining the mapping between one or more relational tables and an XML file. After you have created the mapping, you can generate a document access definition (DAD) script, which can be run by the DB2 XML Extender to either compose XML files from existing DB2 data, or decompose XML files into DB2 data.

7.7 Relational database tools

WebSphere Studio has a Data perspective that gives you useful tools for working with relational databases. Numerous database vendors are supported.

The relational database tools include views, wizards, editors, and other features that make it easy for you to develop and test the database elements of applications. The Data perspective has three views to manage database resources.

**DB Servers view**

With the DB Servers view, you can connect to existing databases and view their designs and existing objects. You can browse and import the designs to the Data Definition view, where you can extend or modify the designs. You can also run stored procedures and user-defined functions (UDFs) and view the results in the Output view.
Data Definition view
With the Data Definition view, you can work locally and directly with relational data objects. The Data Definition view can hold local copies of existing data definitions imported from the DB Servers view, designs created by running data description language (DDL) scripts, or new designs that you have created directly using WebSphere Studio. You can also export data definitions to another database that is installed either locally or remotely. One benefit of the Data Definition view is that you can work on database objects and definitions even when you are not connected to a database server.

DB Output view
With the DB Output view, you can see the messages, parameters, and results related to the database objects that you work with. The DB Output view includes an actions list on the left side of the view and several tabbed pages on the right side for viewing messages, parameters, and results.

7.7.1 Databases and schemas
In addition to importing existing data definitions and exporting data definitions, you can also create new data definitions directly in the Data Definition view. With the Data perspective, you can use wizards to define a database, define a schema, and create a connection to a database.

7.7.2 Tables and views
After you define a database and schema, you can create and work with tables and views for that database. The Data perspective includes wizards and dialogs for defining tables, columns, primary keys, foreign keys, relational views, and data object filters.

7.7.3 SQL statements
With the SQL wizard and SQL query builder, you can create SQL statements. The wizard supports the creation of simple SELECT, INSERT, UPDATE, and DELETE statements, while the SQL query builder supports more complex statements.

7.7.4 DB2 routines
With the Data perspective, you can build DB2 stored procedures and user-defined functions (UDFs). Wizards for creating stored procedures and UDFs provide easy-to-use visual interfaces that guide you through the steps to create routines (stored procedures or UDFs) DB2.
With the Data Definition view you can create routines and work with the source code before building them on the DB2 database server. Existing routines can be viewed and manipulated with the DB Servers view.

With the Editor view, you can view and change the source code and SQL statements that make up stored procedures and UDFs. With the Output view, you can see the messages, parameters, and results related to building and running stored procedures and UDFs.

### 7.7.5 SQLJ files

With the SQLJ File wizard and related features, you can create SQLJ files and SQLJ-enabled projects. SQLJ files are Java files that can contain SQL statements directly within Java code, using special SQLJ syntax. SQLJ-enabled projects can contain SQLJ files. WebSphere Studio will translate SQLJ statements into Java and bind the SQL statements to a DB2 database.

### 7.8 Java development tools (JDT)

WebSphere Studio’s Java development tools (JDTs) are a set of extensions to the workbench that allow you to edit, compile, and run Java programs. The tools consist of a number of perspectives, editors, and views.

#### 7.8.1 Java perspectives

There are multiple perspectives suited for Java development.

**Java perspective**

The Java perspective is the primary perspective for working with Java projects. Its primary views are:

- The Package Explorer view

  This view shows the Java element hierarchy of the Java projects in the workbench. It provides you with a Java-specific view of the resources shown in the Navigator. The element hierarchy is derived from the project's build paths.

  The source folders and referenced libraries for each project are shown in the tree.

  From this view, you can open and browse the contents of both internal and external JAR files. Opening a Java element inside a JAR opens the Class File Editor, and if there is a source attachment for the JAR file, its corresponding source is shown.
Chapter 7. WebSphere Studio V5 overview and architecture

7.8.2 Java Editor

The Java Editor provides specialized features for editing Java code. Associated with the editor is a Java-specific Outline view, which shows the structure of the active Java compilation unit. It is updated as the user edits the compilation unit.

The editor includes the following features:

- Syntax highlighting
- Content/code assist
- Code formatting
- Import assistance
- Quick fix - The editor proposes fixes to common errors
- Integrated debugging features
The Java Editor can be configured to either show an entire compilation unit or a single Java element only.

The most common way to invoke the Java Editor is to open a Java file from the Navigator or Package explorer using pop-up menus or by clicking the file (single or double-clicking depending on the user preferences). You can also open the editor by opening Java elements, such as types, methods, or fields, from other views.

### 7.8.3 Java builder

The Java builder builds Java programs using a compiler that implements the Java Language Specification. The Java builder can build programs incrementally as individual Java files are saved. Problems detected by the compiler are classified as either warnings or errors. The existence of a warning does not affect the execution of the program; the code executes as if it were written correctly.

The Java compiler can create Java class files even in the presence of compilation errors. However, in the case of serious errors (for example, references to inconsistent binaries, most likely related to an invalid build path), the Java builder does not produce any class files.

### 7.8.4 Refactoring support

The goal of Java program refactoring is to make system-wide code changes without affecting the behavior of the program. The Java tools provide assistance in easily refactoring code. Numerous refactoring functions exist including: Extracting, renaming, inlining, converting local variables to fields, pushing members up to superclasses, and pulling members down.

When performing a refactoring operation, you can optionally preview all of the changes resulting from a refactoring action before you choose to commit them. When previewing a refactoring operation, you will be notified of potential problems and will be presented with a list of the changes the refactoring action will perform. If you do not preview a refactoring operation, the change will be made in its entirety and any resultant problems will be shown. If a problem is detected that does not allow the refactoring to continue, the operation will be halted and a list of problems will be displayed.

Refactoring commands are available from the context menus of several Java views (for example, Package Explorer, Outline) and editors.
7.8.5 Debugger

The debugger enables you to detect and diagnose errors in programs running either locally or remotely. It allows you to control the execution of a program by setting breakpoints, suspending launched programs, stepping through the code, and examining the contents of variables.

The debugger has a client/server design so you can debug programs running remotely on other systems in the network as well as programs running locally on a workstation. The debug client runs inside the workbench. The debugger server runs on the same system as the program you want to debug. This could be a program launched on your workstation (local debugging) or a program started on a computer that is accessible through a network (remote debugging).

Among the views in the Debug perspective are:

- Breakpoints: The Breakpoints view lists all the breakpoints you have set in the workbench projects. You can double-click a breakpoint to display its location in the editor. In this view, you can also enable or disable breakpoints, delete them, or add new ones.
  
  This view also lists Java exception breakpoints, which suspend execution at the point where the exception is thrown. You can add or remove exceptions.

- Console: This view shows the output of a process and allows you to provide keyboard input to a process. The console shows three different kinds of text, each in a different color: Standard output, standard error, and standard input.

- Debug: This view allows you to manage the debugging or running of a program in the workbench. It displays the stack frame for the suspended threads for each target you are debugging. Each thread in the program appears as a node in the tree. It displays the process for each target you are running. If the thread is suspended, its stack frames are shown as child elements.

- Display: This view displays the result of evaluating an expression in the context of the current stack frame. You can evaluate and display a selection either from the editor or directly from the Display view.

- Expressions: Data can be inspected in the Expressions view. You can inspect data from a scrapbook page, a stack frame of a suspended thread, and other places in the debugger. The Expressions view opens automatically when the Inspect command is activated.

- Variables: This view displays information about the variables in the currently-selected stack frame.
7.8.6 Scrapbook

The JDT contributes a scrapbook facility that can be used to experiment and evaluate Java code snippets without building a complete Java program. Snippets are edited and evaluated in the Scrapbook Page Editor, with resultant problems reported in the editor.

From the Java Scrapbook Editor, you can select a code snippet, evaluate it, and display the result as a string. You can also show the object that results from evaluating a code snippet in the debugger's expressions view.

7.8.7 Java search

The Java searching support allows you to find declarations, references, and occurrences of Java elements (packages, types, methods, fields). Searching is supported by an index that is kept up to date in the background as the resources corresponding to Java elements are changed. The Java search operates on workspaces independent of their build state. For example, searches can be conducted when auto-build is turned off.

Java searches can be initiated on Java elements to find references, declarations, implementors, read access to a field, write access to a field, and occurrences of the element in its file. The scope of a search can be defined as workspace, hierarchy, or a user-defined working set.

7.8.8 Javadoc

The JDT uses the Javadoc command (typically available in JRE distributions) to generate Javadoc documentation from source files. A Generate Javadoc wizard also can be used to create and export Javadoc documentation.

7.8.9 Visual Editor for Java

The Visual Editor for Java (shown in Figure 7-4 on page cccxlv) provides an interface to help with the design and construction of a Graphical User Interface (GUI) for a Java application or applet. It is based on the JavaBeans component model and supports visual construction using either the Abstract Window Toolkit (AWT) or Swing.

You can use the Visual Editor for Java to create a class that includes pre-defined Java beans from the palette. This class can be a runnable class or it can be a Java bean that you intend to include in another class.

The Visual Editor combines this with a design canvas that lets you lay out and preview any visual Java beans. The Visual Editor is designed to work with .java
source, letting you edit the source and the design simultaneously. It does not have its own perspective.

![Visual Editor for Java](image)

**Figure 7-4  Visual Editor for Java**

**Design and source windows**

The Visual Editor for Java is divided into two windows, one showing the Java beans in a design window, and the other showing the associated source. In the design window, Java beans that are AWT or Swing widgets are shown graphically.

The design window is a WYSIWYG surface that lets you compose the GUI you are building, while the source pane shows the contents of the Java file. The design window has a palette on the left that allows you to control the selection of Java beans.
As you modify the Java beans in the design window, the source is updated. Conversely, as you change the source files, the design window is updated to reflect the changes you make. This round tripping of the design and source windows is designed so that the Visual Editor for Java can be used not only as a tool to generate code, but as an editor to show the effect of source code modifications during development. After you make changes to a .java file source in another editor, the changes will be reflected in the design window of the Visual Editor.

The Visual Editor for Java, by default, shows the design window above the source window on a split pane. You can change the default preference so that the design and source windows have notebook tabs, allowing you to click back and forth between them.

**Properties view and Java Beans view**

There are two additional views that are not included in the standard Java perspective but are used by the Visual Editor for Java. These are the Properties view and the Java Beans view, which are automatically activated when the Visual Editor opens.

The Java Beans view is a tree that shows the components used by the GUI. The Properties view shows the properties for the selected Java bean. The design window and the Java Beans view show the Java beans that are being used by the class being composed.

### 7.9 Enterprise JavaBean (EJB) tools

WebSphere Studio provides a specialized environment that you can use to develop and test enterprise beans that conform to the distributed component architecture defined in the Sun Microsystems Enterprise JavaBeans (EJB) specification. It also supports extended functionality for the WebSphere Application Server. EJB tools are not included in Site Developer.

The EJB support consists of the J2EE perspective and tools for:

- Importing existing EJB JAR files
- Creating enterprise beans and access beans
- Building data persistence into enterprise beans
- Generating deployment code
- Validating enterprise beans for specification compliance
- EJB and Java UML visual editing
7.9.1 J2EE perspective

All of the EJB tools are accessible from the J2EE perspective. This perspective provides a layout with the most commonly used actions, views, and wizards for J2EE and EJB development.

7.9.2 Creating enterprise beans and access beans

The EJB tools help you create enterprise beans (either with or without inheritance) such as session beans, container-managed persistence (CMP) entity beans, bean-managed persistence (BMP) entity beans, or message-driven beans. The EJB Deployment Descriptor Editor helps you set deployment descriptor and assembly properties for enterprise beans.

You can also accomplish complementary enterprise bean development activities, such as writing and editing business logic, importing or exporting enterprise beans, and maintaining both enterprise bean source code and generated code using the built-in Java development tools, along with the team and versioning capabilities of WebSphere Studio.

7.9.3 Creating access beans

You can create access beans, which are Java bean wrappers for enterprise beans. Access beans are typically used by client programs, such as JSPs, servlets, and sometimes even other enterprise beans.

7.9.4 Building data persistence into enterprise beans

The EJB mapping tools help you map entity enterprise beans to back-end data stores, such as relational databases. There is support for top-down, bottom-up, and meet-in-the-middle mapping development. You can also create schemas and maps from existing EJB JAR files.

7.9.5 Generating deployment code

The EJB tools generate the deployment classes that allow beans to run on an EJB server. A wizard can be launched from the selected EJB project or module in the J2EE Hierarchy view. This wizard provides a list of the enterprise beans that you can deploy. These tools mask the complexities normally associated with creating deployment classes, such as generating RMI-over-IIOP stubs and EJB container-specific deployment code.

The tools support session beans, CMP entity beans, BMP entity beans, and message-driven beans. They also allow you to create relational database tables.
for CMP entity beans. After the deployment code is generated, you can export enterprise beans to a JAR or EAR file for installation on an EJB server, such as the WebSphere Application Server.

7.9.6 Validating enterprise beans and access bean code

The EJB tools automatically and seamlessly validate that enterprise bean code is consistent and that it conforms to the rules defined by the Enterprise JavaBeans specification. Code verification occurs whenever an enterprise bean or its properties are changed. Errors and warnings are displayed in the Tasks view of the workbench. The corresponding file also displays a warning icon.

The EJB tools also automatically validate that access beans are constructed correctly and that they are consistent with their associated enterprise beans. Code validation occurs whenever you create or edit access beans.

7.9.7 EJB and Java UML visual editing

With the features of EJB and Java Unified Modeling Language (UML) visual editing, you can visualize and edit the artifacts of J2EE applications using UML notation within a single, tightly integrated Java development environment.

You can visualize and edit EJB components of a project from the Class Diagram Editor window. You can create and populate new UML class diagrams to visualize the structure and relationships for EJB components and contained classes. You can create new EJB components, add EJB relationships, show EJB operations in class shapes, and edit and rename EJB components from an open UML class diagram.

7.10 Web services development tools

WebSphere Studio provides a toolbox for discovering, creating, and publishing Web services that are created from Java beans, Document Access Definition Extender (DADX) files, EJBs, and Universal Resource Locators (URLs). You can also use Web service tools to create a skeleton Java bean and a sample application from a Web Services Description Language (WSDL) document.

The capabilities run from simply using an existing Web service in an application, to full development, testing, and publishing of your own Web services.
WebSphere Studio provides the following support to assist with Web services development:

- **Discover**: Browse Universal Description Discovery and Integration (UDDI) registries or Web Services Inspection Language (WSIL) documents to locate existing Web services for integration.

- **Create or transform**: Create Web services from existing artifacts, such as Java beans, enterprise beans, URLs that take and return data, DB2 XML Extender calls, DB2 stored procedures, and SQL queries.

- **Build**: Wrap existing artifacts as SOAP and HTTP GET/POST accessible services and describe them in WSDL. The Web services wizards assist you in generating a Java client proxy to Web services described in WSDL and in generating Java bean skeletons from WSDL.

- **Deploy**: Deploy Web services into the WebSphere Application Server or Apache Tomcat test environments using server tools.

- **Test**: Test Web services running locally or remotely in order to get instant feedback.

- **Develop**: Create Web service client applications.

- **Publish**: Publish Web services to a UDDI V2 Business Registry.

Web services support is provided by the following tools:

- **Web Services Client wizard**: Used to create a Java client for a deployed Web service and to test the Web service.

- **Web Services DADX Group Configuration wizard**

- **Web Services wizard**: Used to create, deploy, test, and publish Web services based on artifacts such as Java beans, URLs, enterprise beans, and DADX files.

- **Unit Test UDDI wizard**: Used to install and configure, or remove, a private UDDI registry.

- **Java Beans for XML Schema wizard**: Used to generate Java beans from schema.

- **IBM Web Services Explorer**: Assists in discovering and publishing Web service descriptions.

### 7.10.1 Creating Web services

The following are approaches to creating Web services:

- Transform an existing application (bottom-up) into a Web service.
Transforming an existing application into Web services includes the generation of service wrappers to expose the business functionality.

- Use an existing service definition, WSDL, to generate a new application (top-down).

  Generating a new application includes using a specific programming language and model.

- Combine an existing group of already generated Web services to provide a new integrated function (multiple services) using WebSphere Studio Application Developer Integration Edition workflow technologies.

**Creating Web services from existing applications (bottom-up)**

Web services are created through the use of the Web Service wizard. This wizard allows you to:

- Create a new Web service.
- Create a sample application.
- Configure it for deployment and deploy it. The server can be a test environment server in WebSphere Studio or an external application server. WebSphere, IBM SOAP, or Apache Axis can be specified for the runtime.
- Launch the Web Services Explorer so you can publish the Web service to a UDDI registry.

For bottom-up development (the most common case) the following data structures can be used to build a Web service:

- Java bean
- EJB (not available in Site Developer)
- Document access definition extension (DADX)

DADX is an XML document format that specifies how to create a Web service using a set of operations that are defined by DAD documents and SQL statements. A DADX Web service enables you to wrap DB2 XML Extender or regular SQL statements inside a standard Web service. The DADX file defines the operations available to the DADX runtime environment, and the input and output parameters for the SQL operation.

You can use XML tools to create DAD and DADX files or write DADX files using any text editor. After you create a DADX file, use the Web Services DADX Group Configuration wizard to configure database connections. Then use the Web Service wizard to generate WSDL documents, deployment descriptors, client proxy, property mappings, deployment mappings, and test the DADX Web service, in preparation for publishing to the UDDI registry.

- URL
You can create a new Web service that directly accesses a servlet running on a remote server.

▶ ISD

An ISD file is a Web service deployment descriptor and provides information to the SOAP runtime about the service that should be made available to clients, for example, URI, methods, implementation classes (Java bean, EJB), serializers, and deserializers. ISD files are concatenated into the SOAP deployment descriptor, dds.xml.

The ISD Web service enables you to redeploy beans and EJBs without having to re-specify all of the configurations and mapping information. The ISD stores information such as EJB JNDI, EJB home name, URI, methods, classes, scope, and type mappings.

**Creating Web services from existing WSDL (top-down)**

For top-down development, WebSphere Studio provides these functions:

▶ Java bean skeleton from WSDL

The Web Service wizard assists you in creating a skeleton Java bean from an existing WSDL document. The skeleton bean contains a set of methods that correspond to the operations described in the WSDL document. When the bean is created, each method has a trivial implementation that you replace by editing the bean.

▶ Java bean from XML schema (XSD)

The Web services tools support the generation of Java beans from an XML schema. Using these beans, you can create a Java bean Web service.

▶ EJB skeleton from WSDL

Create a skeleton EJB that contains a set of methods that correspond to the operations described in the WSDL document. When the EJB is created, each method has a trivial implementation that you replace by editing the bean. This function is not available in WebSphere Studio Site Developer.

### 7.10.2 Using Web services (developing clients)

To assist in the development of Web service clients, WebSphere Studio provides the Web Service Client wizard and the Web Service Data Model wizard.

**Web Service Client wizard**

The Web Service Client wizard assists you in generating a proxy Java bean and a sample application. The sample Web application demonstrates how to use the proxy bean in a client program. Note that the proxy and sample can also be generated in bottom-up and top-down approaches for testing of the generated
Web service. The invocation can be implemented either *statically* or *dynamically*.

**Web Service Data Model wizard**
The Web Service Data Model wizard provides a way of using Web services to populate certain Swing components. It takes you through the steps of creating a model for Swing components, which populate themselves through a model. The Swing components supported are the JTable and the JList. The wizard helps you generate a TableModel or ListModel containing the code required to query a Web service and map the returned data to the requirements of that respective Swing component.

The Web Service Data Model wizard follows a simple three-step process of integrating Web services into Swing components through their models:

1. Select a Web service client.
2. Select an output from the many services the Web service client provides. Optionally, provide input parameters for that service.
3. Map the API of the selected output to the Swing component.

### 7.11 Portal tools

The Portal Toolkit plug-in for WebSphere Studio provides support for portlet development. The Toolkit is supported for both the Site Developer and Application Developer editions and is included in the package with the Application Developer.

The toolkit provides:

- An Integrated WebSphere Portal test server
- Portlet projects, in which you can create basic portlets, JSP portlets, servlet invoker portlets, XSL portlets, and multi-device (MVC) portlets
- A wizard to create JSR 168 Portlets and WebSphere Portal API portlets, also enabling the use of JSF components in both of these portlet types and Struts with WebSphere Portal API portlets
- Drag/drop components for Click-to-Action and People Awareness features (WebSphere Portal features)
- Visual portlet editing in Page Designer
- Portlet application examples for enterprise applications

The Portal Toolkit is available for download at:
7.12 Team collaboration

A plug-in called Team Support is provided by Eclipse to enable team collaboration and functions such as sharing resources, version control, problem tracking, change management, build and deployment management, access control, and process management.

The Team Support plug-in lets the environment take advantage of external Software Configuration Management (SCM) systems through the use of adapters. Each SCM product requires a specific adapter that should be supplied by the product vendor. WebSphere Studio includes adapters for Concurrent Versions System (CVS), for Rational ClearCase LT (not supplied with WebSphere Studio Site Developer), and for Software Configuration and Library Manager (only included in WebSphere Studio Enterprise Developer).

The Eclipse platform is based on the concept of letting users create and edit resources in the file system. The Team Support plug-in defines an API that allows adapters to integrate the function of a versioning and configuration management repository.

Multiple repository providers can coexist peacefully within the platform. In fact, it is even possible to have different client implementations for the same repository installed. For example, one could install a CVS client designed for experts and a different one for novice users.

7.12.1 Concurrent Versions System (CVS)

CVS is an open source team resource management system that uses a branch model to support multiple courses of work that are somewhat isolated from each other but still highly interdependent. This model allows individuals to work on a team project, share their work with others as changes are made, and access the work of others as the project evolves.

Users can commit their work to the CVS repository, synchronize or catch up their local resources with resources in the repository, compare and merge local and repository resources, plus many more advanced functions.

7.12.2 Rational ClearCase LT

Rational ClearCase LT is provided with WebSphere Studio Application Developer and with the more advanced configurations. It helps to automate the tasks required to write, release, and maintain software code by providing the essential
functions of version control, workspace management, process configurability, and build management. It automates many of the error-prone tasks associated with controlling software development projects.

ClearCase incorporates Unified Change Management (UCM), a best practices process for managing change at the activity level and controlling workflow. UCM can be applied to projects “out-of-the-box”, enabling teams to get up and running quickly, or it can be replaced with other processes that are already in place.

ClearCase provides support for parallel development, allowing multiple developers to design, code, test, and enhance software from a common code base. It includes automatic branching and snapshot views. Snapshot views support a disconnected use model for working away from the office. All changes since the last snapshot are automatically updated once you are connected again.

7.12.3 Software configuration and library manager (SCLM)

WebSphere Studio Enterprise Developer has a Software Configuration and Library Manager (SCLM) plug-in that provides SCLM services to projects being developed in the workstation environment. The SCLM Repository view gives you full menu-driven access to SCLM functions.

SCLM is a proven and reliable tool to help you manage source and object code for z/OS software development projects. This tool helps you manage software changes and analyze the impact of changes.

SCLM has a library manager and a configuration manager. The library manager manages a project’s source code. The configuration manager compiles source code, links the resultant components, and integrates those software components.

7.13 Integrated debuggers

WebSphere Studio includes a variety of debuggers that enable you to detect and diagnose errors in programs running either locally or remotely. The debuggers let you control the execution of a program by setting breakpoints, suspending execution, stepping through code, and examining the contents of variables.

You can debug live server-side code as well as programs running locally on a workstation. The debugger includes a Debug view that shows threads and stack frames, a Processes view that shows all currently running and recently terminated processes, and a Console view that lets you interact with running processes. There are also views that display breakpoints and let you inspect variables.
7.13.1 Java development tools debugger

The Java debugger has a client/server design so that it can be used to debug programs that run locally (on the same workstation as the debugger) or remotely (on another computer on the network).

Local debugging is the simplest and most common kind of debugging. After you have finished editing and building a Java program, you can launch the program on your workstation. Launching the program in this way will establish a connection between the debugger client and the Java program being launched.

The client/server design also allows you to launch a Java program on a remote server and debug it from the development machine. This is particularly useful when you are developing a program for a device that cannot host WebSphere Studio. It is also useful when debugging programs on dedicated machines such as Web servers.

7.13.2 Server-side JavaScript debug adapter

The server-side JavaScript debug adapter enables you to detect and diagnose errors in JavaScript that is running locally or remotely on a WebSphere Application Server.

7.13.3 Active Script debugger

The Active Script debugger enables you to detect and diagnose errors in code that contains Active Script (such as client-side JavaScript) or Active Script objects.

With the Active Script debugger, you can debug JavaScript files (extension .js) and VisualBasic files (extension .vbs). You can also debug JavaScripts and VisualBasic scripts that are embedded in HTML and JSP files.

7.13.4 SQL stored procedure debugger

The SQL stored procedure debugging plug-in enables you to detect and diagnose errors in SQL procedures that are running on a local or remote DB2 server.

7.13.5 SQLJ debugger

The SQLJ debug adapter allows you to begin a debug session for SQLJ (or Java embedded with SQL). Once you have launched a debug session for SQLJ, you can use the Java debugger to control execution.
7.13.6 Compiled language debugger

The compiled language debugger plug-in enables you to detect and diagnose errors in programs that are running locally or remotely in languages such as C and C++. The debugger contains views and functionality specific to debugging compiled language applications such as:

- Choosing from multiple breakpoint types, such as line, entry, address, watch, and load
- Specifying exception types at which the debugger will stop program execution
- Choosing the way you would like to view source in the editor
- Monitoring storage, storage maps, registers, variables, and expressions

7.14 Profiling and logging tools

WebSphere Studio provides a Profiling and Logging perspective and the Log and Trace Analyzer that enable you to test application performance early in the development cycle. This allows enough time to make architectural changes and resulting implementation changes. This reduces risk early in the cycle, and avoids problems in the final performance tests.

The Log and Trace Analyzer combines the features of the Logging tool (also known as Log Analyzer) and the Profiling tool.

The Logging tool enables you to import various log files as well as symptom databases against which log files can be analyzed and correlated.

The Profiling tool provides the ability to profile applications, to work with profiling resources, and to interact with the applications you are profiling.

The Log and Trace Analyzer provides standalone and plug-in support for new and existing users of the following logging utilities:

- JSR-047 Java Logging APIs
- Commons Logging

7.14.1 Log Analyzer

The Log Analyzer allows you to import and analyze log files (or trace files) from multiple products, as well as determine the relationship or correlation between the events captured by these products.

An interface provides a single point of operation to deal with logs and traces produced by various components of a deployed system. Linking these two sets
of tooling (tracing and logging) helps bridge the gap between problem
determination and debugging of applications and middleware. By capturing and
correlating end-to-end events in the distributed stack of a customer application,
this tool allows for a more structured analysis of distributed application problems.
The tool also makes it easier and faster for a person to debug and resolve
problems within a system.

The Log Analyzer provides several correlation methods for events in a single
product log:

- Sequential correlation: The ability to sort the events in a log by various fields
  contained in the events (for example, timestamp)
- Associative correlation: The ability to filter the events displayed in a log by the
  values in various fields contained in the events (for example, thread ID)

The Log Analyzer maps the currently supported log format into a common event
model called *Common Base Event*, which is a standard for events among the
different types of enterprise applications. This allows the analyzer to use a
common format for any log records from any supported log files. The parsers
provided with the Log Analyzer map the log records from their current output
format to this common model.

The Log View, as shown in Figure 7-5, is used for working with an imported log.
The Log Analyzer can correlate several logs and create log thread interaction diagrams that display interactions among log events that occur on different threads of an application’s execution.

Figure 7-6  Sequence diagram showing correlation between access log and error log

The Log Interactions view, as shown in Figure 7-6, shows the correlation between instances of log records. Those interactions can be correlated based on various correlation schemas according to your preference.

Sequence Diagram views can be opened from within the Profiling and Logging perspective. You can view the correlation from different levels of abstraction, starting with log interactions, through interactions among processes, up to host interactions across a network. The importance of this multilevel data presentation becomes obvious for the monitoring of e-business applications.

There are different types of log file correlations, depending on the correlation schemas that you select:

- Correlation by time
- Correlation by URLs and time
Correlation by application IDs and time
Correlation by URLs, application IDs, and time

**Symptom database**
A symptom database is an XML file of symptoms, string match patterns, associated solutions, and directives used in the analysis of event and error messages that may occur in a log.

Log records can be analyzed using the symptom database to interpret known events and error conditions. The symptom database may contain detailed information on error resolution and event significance.

Symptom databases for WebSphere Application Server products are provided. You can also import symptom databases from a local or a remote host.

### 7.14.2 Profiling tool

The Profiling Tool consists of the Profiling and Logging perspective and a number of views. It enables you to profile applications, to work with profiling resources, to interact with the applications you are profiling, and to examine applications for performance and memory usage problems.

The Profiling Tool collects data related to the Java program's runtime behavior and presents this data in both graphical and tabular views. All these facilities help you to visualize and understand the program's execution as well as to explore patterns of program behavior.

The Profiling Tool also enables you to see which operations take the most time and help you to find and solve memory leaks. You can easily identify repetitive execution behavior and eliminate redundancy. In short, the Profiling Tool is broadly useful for performance analysis and for gaining a deeper understanding of a Java program.

To profile an application, the IBM Agent Controller must be installed and running on the host where the application under test resides.

**Profiling architecture**
WebSphere Studio provides the developer with a set of tools to allow for early analysis of performance-related issues in Java applications. The profiling tools can be used to gather performance information on applications that are running:

- Inside an application server, such as WebSphere
- As a standalone Java application
- On the same machine as the workbench
- On a remote machine
In multiple JVMs

Using filters, you can focus on classes that you are interested in and omit tracing for others.

Profiling creates a number of different graphical and tabular views of a Java program’s runtime behavior, and simplifies identifying and diagnosing performance-related problems.

The basic architecture of the profiling tools involves the Java Virtual Machine (JVM) where the application is running, an agent running inside the JVM capturing profiling information, an agent controller that controls the agent and retrieves profiling information, and the performance analyzer inside WebSphere Studio. The relationships between the components are shown in Figure 7-7.

The agent runs inside the JVM and uses the Java Virtual Machine Profiler Interface (JVMPI) to interface with the JVM.

![Figure 7-7  WebSphere Studio profiling architecture](image)

Each application process being profiled may have a profiling agent running within it. Thus, the Profiling Tool makes it possible to interactively profile these applications from a single workbench.

There are two agents available for you to select: Java Profiling Agent and J2EE Request Profiler.
Java Profiling Agent
The Java Profiling Agent collects data within the boundaries of a single JVM execution space. The agent is attached to a JVM in which the profiled application runs. Profiling focuses on the level of an agent or process and provides the following types of sequence diagrams:

- Object interactions
- Class interactions
- Thread interactions

J2EE Request Profiler
The J2EE Request Profiler is an agent that resides within the application server process for the purpose of collecting data from the interception points of the e-business application's requests. The J2EE Request Profiler uses the Agent Controller to externalize this data so that it can be rendered by the various views provided by the Profiling and Logging perspective of the workbench.

Profiling of distributed e-business applications that run either within the boundaries of one machine or on a cluster of machines is accomplished by using either a single instance or several instances of the J2EE Request Profiler. Each J2EE Request Profiler is responsible for collecting profiling information for the application server process it resides within. Attaching to a number of J2EE Request Profilers on different application servers enables the collection of profiling data across these servers.

You have a choice as to the point in time within the application's execution period at which to attach. You can also select the J2EE Request Profiler that you attach to. Tracing of the application starts from the moment you attach to the J2EE Request Profiler and start monitoring the application. As the execution of the application crosses the boundaries of a host, the remote discovery mechanism causes the attachment to other instances of the J2EE Request Profiler which represent remote WebSphere Application Servers. These WebSphere Application Servers host servlets and enterprise beans that participate in the application execution.

The J2EE Request Profiler collects data from requests arriving in EJB containers as well as Web containers. This data collection mechanism enables the creation of sequence diagrams, which represent interactions among servlets, JSPs, and enterprise beans, while ignoring other artifacts of the application infrastructure that do not represent the business logic of the application. The collected data enables the creation of a variety of different diagrams, which are defined for specific levels of the profiling hierarchy (monitors, hosts, processes, and agents).
Sequence diagram views
The Sequence Diagram views of the Profiling perspective offer the following diagram types:
- Host interactions
- Process interactions
- Thread interactions
- Object interactions
- Class interactions

Statistical views
The tool provides four statistical views that display details about a profile session:
- Package statistics
- Class statistics
- Method statistics
- Instance statistics

Execution Flow view and table
The Execution Flow view and table both show a representation of the entire program execution. In the view, the threads of the program fit horizontally, and time is scaled so that the entire execution fits vertically. In the table, the threads are grouped in the first column and time is recorded in successive rows.

Method Invocation view and table
The Method Invocation view shows a representation of the entire course of a program's execution and also provides the ability to navigate through the methods that invoked the selected method. The view uses the same graphical metaphor as the Execution Flow view and the Method Invocation table contains the same information as the Execution Flow table.

Object references table
The Object References table displays references to or from a set of objects. This is useful to study data structures, to find memory leaks, and to find unexpected references.

7.15 Component test tools
The Component Test perspective provides a framework for defining and executing test cases. The basic framework supports three sorts of test cases, based on their different schedulers: Manual, Java, and HTTP. You can also create report generators to work with the data returned by an executed test case.
The component test tools allow you to:

- Define manual test cases that automate a tester's to-do list.
- Define HTTP test cases that automate requests against a Web site.
- Define Java test cases that implement the JUnit framework to automate Java method calls.
- Run test cases locally or remotely using the IBM Agent Controller.
- Track execution results as the test case executes.
- Generate reports on test case information.
- Define new report generators.

The perspective can be used by developers to test their own code, or by testers to coordinate project-wide testing efforts.

**JUnit**

JUnit is an open source testing framework for Java that is used to develop and execute repeatable unit tests in Java. The Java Development Tools (JDTs) include a plug-in that integrates JUnit into the Java IDE. The JUnit plug-in allows you to define regression tests for code and run them from the Java IDE.

WebSphere Studio has a wizard to help you create JUnit test cases and test suites. A test suite is a group of test cases that are run together.

The JUnit plug-in defines an extension point that allows you to observe JUnit test runs and implement a reporting scheme in your plug-in. This extension point is useful if you are implementing test or build tools that must report the results of automated test runs.

**Report generators**

You can run reports on your execution results, and on any other test case elements for which you have report generators configured. Each report generator may have different valid selections. The default report generator provided with Component Test is for comparing execution results, so you must select at least two execution result elements before the report becomes available on their pop-up menu. It creates a graph in SVG markup that compares the results you selected.

### 7.16 Server tools for testing and deployment

The server tools feature of WebSphere Studio provides runtime environments where you can test JSP files, servlets, HTML files, enterprise beans, and Java...
classes. A universal test client (UTC) is provided to test EJB modules and Web services. It also provides the capability to configure other local or remote servers for integrated testing and debugging of Web and EJB applications.

You can use the server tools to test and publish resources from the following types of projects:

- Web projects, which typically contain JSP files, HTML files, servlets, and Java beans. Web projects can also contain Web services that can be tested.
- Enterprise Application projects, which may contain Java Archive (JAR) files or Web Archive (WAR) files, or both, and pointers to other Web or EJB projects.
- EJB projects, which contain Enterprise Java beans.
- J2EE Application client projects.

After testing an application locally, you can use the server tools to publish the application either locally or remotely to another machine.

**Supported runtime environments**
The server tools allow you to test applications in different local or remote runtime environments. Where indicated, you can also use the tools to publish to these environments:

- WebSphere test environment: All product configurations provide one or more versions of the WebSphere Application Server runtime environment, which you can use to test applications directly from the development environment. Each test environment provides all the function of the full runtime environment, but eliminates dependencies on network connections.
- WebSphere Application Server: You can also test on and publish to one or more separately installed versions of WebSphere Application Server that reside either locally or on a remote machine. If the test server is remote, you must install Agent Controller on the remote machine.
- Apache Tomcat: You can test on the Apache Tomcat runtime environment, running locally. With Tomcat, you can only test Web projects that contain servlets and JSPs (not available with WebSphere Application Server - Express).
- TCP/IP monitoring server: This is a simple server that monitors all the requests and responses between the Web browser and an application server. It also monitors TCP/IP activity. You can only test projects locally in this runtime environment. You cannot publish projects to the TCP/IP Monitoring Server.
- Static Web publishing server: You can publish static Web projects to an HTTP Web server.
- J2EE publishing server: You can publish J2EE projects to a generic J2EE publishing server.

- WebLogic V6.1 and V7.0: You can create, test, and deploy J2EE applications to a WebLogic server (not available with WebSphere Application Server - Express).

### Server definitions

The server tools use *servers* and *configurations* to test and publish projects. Servers are definitions that identify where you want to test projects. Server configurations contain setup information. You can either have the development environment create the servers and configurations automatically for you, or you can create them using a wizard.

### Server perspective

WebSphere Studio provides a server perspective that allows you to manage servers and server configurations. A Server Configuration view allows you to create or delete servers and server configurations. A Servers view gives you the ability to stop, start, and restart servers; launch a server in debug or profiling mode; and publish. The Console view allows you to monitor runtime messages.

### 7.16.1 WebSphere test environment

The server tools contain the complete runtime environment of the WebSphere Application Server. This environment is called the WebSphere test environment. The test environment includes a local copy of the full IBM WebSphere Application Server Advanced Single Server Edition for Multiplatforms V4 AES and WebSphere Application Server V5 runtime environments, where you can test Web projects, EJB projects, Java application client projects, and enterprise application projects.

The test environment offers the following benefits:

- Standalone all-in-one testing
- No dependency on WebSphere Application Server installation or availability
- No dependency on an external database even when entity bean support is required
- Provides the ability to debug live server-side code
- Supports configuring multiple Web applications
- Supports multiple servers that can be configured and run at the same time
- Provides access to the profiling feature
- Provides the ability to version server configurations
- Provides a Universal Test Client where you can test enterprise beans and Java classes
- Provides access to the WebSphere Application Server administrative console
- Provides the ability to automatically create tables and data sources for testing CMP beans
- Supports sharing servers with multiple development clients
- Supports hot-code replace in the V5 WebSphere test environment

When you install WebSphere Studio, you indicate which versions of the WebSphere test environment you wish to use for testing. Table 7-2 shows you the default server configuration that is installed for each WebSphere Studio configuration.

**Table 7-2  Integrated application server test environments**

<table>
<thead>
<tr>
<th>Application Server</th>
<th>Express 5.1</th>
<th>Site Developer 5.1.2</th>
<th>Application Developer 5.1.2</th>
<th>Integration Edition 5.1</th>
<th>Enterprise Developer 5.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Express Application Server 5.0.2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Express Application Server 5.1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WebSphere Application Server V4</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WebSphere Application Server V5.0.2</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WebSphere Application Server V5.1</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WebSphere Application Server Enterprise V5</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Integration Server 5.1</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
7.16.2 Application publishing

In WebSphere Studio, publishing involves copying files (projects, resource files, and server configurations) to the correct location for the server to find and use them. In the test environments, these files may already be in the correct location. In some cases, temporary copies of the server configurations may be created. For remote servers the files are either copied to the server (if a shared file system exists) or sent using FTP.

Regardless of which environment you are publishing to, the server automatically copies the files to the correct location. If you publish the projects, the remote server is automatically configured using the server configuration that is associated with it.

The server tools will check if project files on the remote server are synchronized with the development copies. If they are not, the project and the files are automatically updated on the server.

Exporting a project

During application publishing, the configuration file created for the server is used to override the configuration of the server. If you do not want to override the configuration of the server you can export a project to a server and manually configure the application server using the configuration tools provided by WebSphere Application Server.

7.16.3 Universal Test Client

The server tools provide you with a Web-based Universal Test Client (UTC) that allows you to test enterprise beans and other Java objects running in a local or remote server. Using this test client, you can test the local or remote interface methods of enterprise beans. By calling the methods and passing user-defined arguments you can test the methods to ensure that they work correctly.

Using the UTC, you can accomplish the following tasks:

- Find an enterprise bean when you know or do not know the JNDI name.
7.16.4 Server administration from WebSphere Studio

When a WebSphere Application Server is used to test applications from the WebSphere Studio, server administration and operation is typically done by the developer from the workbench. Communications between the workbench and the application server are done via the IBM Agent Controller. The IBM Agent Controller is installed automatically when the application server is installed.

When testing on a WebSphere Application Server from the WebSphere Studio server tools, you have two options for performing server configuration:

- The Server Configuration Editor
- The WebSphere Application Server administrative console

Which one you use depends on the circumstances and the type of server.

**Server Configuration Editor**

You can administer common server tasks using the Server Configuration Editor. You can set and configure server settings, class paths, environment options, Web options, data sources, security, server ports, trace options, EJB container, J2C options, JMS provider options, and applications options. However, note that not all options that you would see in the administrative console are available.

Every server configured by using the Server tools will have its configuration stored in the user's Studio workspace. In the case of the unit test environment, this is the primary configuration file. In the case of a remote server (installed external to WebSphere Studio), this configuration file is used to overlay the current configuration file in the server's AppServer/config directory.

**Administrative console**

The administrative console is a Web browser-based GUI console that is actually an application running on a server. This implies that the server has to be up and running for you to use the console. It is always installed on the server application server, whether the server is part of the unit test environment or a standalone server. You can access the console by selecting the server in the Servers view and selecting Run Administrative Console, or directly by entering the URL from a Web browser.
Configuration changes made using the administrative console are stored in the AppServer/config directory.

**Note:** If you are running WebSphere in the unit test environment, you can use the Server Configuration Editor and the administrative console interchangeably. If you are running WebSphere as a standalone server, you should only use one method or the other. Any changes made by the administrative console will be lost the next time changes to the configuration are made from the Server Configuration Editor and the server is restarted from Studio.

### 7.17 Business integration tools

Business integration tools are packaged with WebSphere Studio Application Developer Integration Edition and WebSphere Studio Enterprise Developer configurations.

The tools are built to support a services-oriented architecture, which decreases development complexity by representing all objects, components, applications, and processes as services (based on open standards). These services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Internet, automating a wide range of business functions, including existing IT assets. Using this type of architecture, developers can create and interact with all software components without regard to their underlying implementation.

With business integration tools you can:

- Reduce the time and complexity normally required to integrate with back-end systems by rapidly building Java Connector Architecture (JCA) based application adapters.
- Decrease development complexity by using services (based on open standards) to create and interact with all software components.
- Easily compose and choreograph application interactions, dynamic process flows among J2EE components, Web services, existing applications, and human activities.
- Take advantage of the latest innovations in J2EE technology such as business rule beans, shared work areas, transactional coordination, and compensation support.
- Optimize and simplify J2EE application development with best practices, templates, code generation, and the most comprehensive development environment in its class.
7.17.1 Services-oriented architecture

A services-oriented architecture decreases development complexity by using services based on open standards to create and interact with software components. Using the service tools and wizards developers can:

- Create services of software components such as Web services, Java beans, stateless session EJBs, and processes.
- Create services that access back-end systems such as CICS, IMS, and HOD 3270.
- Consume SOAP, Java beans, stateless session EJBs, and JCA services.
- Describe services using Web Services Description Language (WSDL).
- Generate service proxies.

7.17.2 Business Integration perspective

The Business Integration perspective provides intuitive drag-and-drop tools to allow you to visually choreograph the sequence and flow of information between services to implement a business process. Using the Process Editor and wizards, developers can:

- Visually compose a service out of one or more existing services.
- Define and transform the processing of information between services.
- Create a process that has within itself other nested processes, to multiple levels.
- Include activities that require human interaction as steps in an automated business process.
- Develop a process from the top down by creating a skeleton process that choreographs the sequence of events in an application.

7.17.3 Human Interaction support

Human Interaction support expands the reach of processes to include activities that require human interaction as steps in an automated business process. Processes involving human interaction are interruptible, persistent (a person may take a long time to complete the task), and resume when the person completes the task. Human Interaction support includes:

- Specialized "Staff" activities that represent human interaction in a process.
- Automatic persistence of process state and metadata when human interaction is involved.
- Dynamic assignment of responsibilities based on existing organizational definitions in LDAP or Microsoft Windows User Registry or a customized mapping plug-in to any directory structure.
- Work lists to notify the designated recipient that her action is required.

### 7.17.4 Event triggering

Event triggering expands the scope of processes by including support for events external to the scope of the process. Event triggering support includes:

- User-customizable activities that are triggered by events outside of the process, including JMS messages or any event that can call a Java API
- Event activities that can be configured to trigger a process when a specific event occurs or used inside of a process to stop and wait for an external event to occur

### 7.17.5 Compensation support

Compensation support increases application functionality by providing transaction "undo" features in long-running processes without tying up resources waiting for all of the steps to complete.

### 7.17.6 Process Debugger

The Process Debugger provides a dramatic increase in programming productivity by providing the user with a visual means of debugging complex processes. The process debugger:

- Provides functionality to step through the process activity-by-activity, including stepping over and into activities
- Seamlessly interacts with the Java Debugger to debug the underlying implementation
- Includes functionality to change the input and output data of an activity while debugging for unit testing

### 7.17.7 Fault handling

Fault handling provides you with an easy and integrated means of performing in-process exception handling in processes. Fault handling support includes an output activity that is dedicated to exception handling and can be used to define a different path in the case of an exception.
7.17.8 Extended messaging support

Extended messaging allows you to quickly create applications that integrate with other systems using a messaging infrastructure. Automated support for Java Message Service (JMS) allows you to focus on business logic instead of complex messaging APIs. JMS support includes:

- Automated support for asynchronous outbound (and inbound) JMS messaging
- Wizards to create the sender and receiver beans which automatically handle the JMS messages

7.17.9 Application integration

The time and complexity normally required to integrate with back-end systems is reduced by using JCA application adapters in conjunction with application processes to integrate packages or existing assets with J2EE applications. To make application integration easier, the following tools are provided:

- Tools and wizards to manage the low-level data handling requirements for JCA connectors
- Process tools to quickly create and deploy customer application adapters
- Service tools to make the adapter usable by other parts of the application
- Development connectors for CICS, IMS and HOD, including JCA tool plug-ins, to provide integrated tool support for function catalogs and definitions
- 3270 tools that can capture the images of 3270 screens and record a screen process

7.18 z/OS support tools

z/OS application development tools are available only in WebSphere Studio Enterprise Developer and provide an interactive, workstation-based environment where you can develop:

- Mainframe applications in HLASM, COBOL, or PL/I
- Workstation-based applications in COBOL or PL/I

The environment allows you to seamlessly edit files with a workstation-based editor and to prepare output on the mainframe. For example, you can:

- Modify source while connected to z/OS or offline.
- Use a local compiler as an efficient way to validate z/OS-based files.
Generate and customize JCL as needed.

Debug code locally or remotely.

Use a workbench interface to submit various kinds of mainframe work (like job streams, TSO commands, CLISTs, and REXX EXECs) and to inspect results.

Mainframe programs can target MVS batch, CICS, or IMS.

This section describes the workstation-based development tools followed by a description of the required tools that need to be installed on the z/OS server so that workstation developers can interact with the z/OS system.

### 7.18.1 z/OS development tools

Enterprise Developer provides two perspectives to work with z/OS resources, the z/OS Systems perspective and the z/OS Projects perspective. Editors are available so that developers can work directly with z/OS resources while being connected to a z/OS system or work in an offline mode to go through code, validate, compile, test, and debug cycles locally on the workstation and thereby use fewer of the more expensive z/OS resources.

#### z/OS Systems perspective

You can use the z/OS Systems perspective to perform MVS-based tasks that are equivalent to the following tasks in ISPF:

- View lists of resources.
- Allocate or delete datasets.
- Create or delete members.
- Edit text.
- Submit JCL.
- Submit TSO commands.
- Review job output.

The perspective includes a view called z/OS Systems that gives you access to z/OS resources. Each z/OS system has a set of properties that identify system ports and other build-related resources such as compiler and linkage-editor data sets.

#### z/OS Project perspective

The z/OS Project perspective allows you to define and work with MVS projects that contain code that resides and runs on the host. You can also create local projects for developing COBOL and PL/I code that resides and runs on the workstation.
**MVS project**

An MVS project can be in an online state where the project is connected to the z/OS system or in an offline state that permits access to only workstation-based files.

As you disconnect from z/OS, you can specify the data sets and members to be transferred to the workstation. When you switch back to online state, the specified files are automatically uploaded to the mainframe, with a confirmation message that keeps you from unintentionally overwriting resources.

When you are working in an MVS project in online state you can access IBM's Software Configuration Library Manager (SCLM) to check data sets into the library system and check them out without leaving the development environment.

Also, when you allocate data sets or create PDS members from within an MVS project, the names are validated against z/OS naming rules to avoid spending the time that would be necessary to process allocation errors on z/OS.

**Local project**

Local projects are not associated with a system definition, as MVS projects are. When you define your first local project, you set values for properties such as compiler options.

**Syntax checking**

Local syntax checking of COBOL or PL/I source files uses a workstation-based compiler and can be used to save processing time and to avoid the expense (if any) that is associated with added z/OS CPU cycles.

Remote syntax checking uses a z/OS-based compiler and lets you do a final syntax check on the target system.

**Debugging**

When you develop COBOL or PL/I code with z/OS application development tools, you use the Debug perspective to handle either of two cases:

- You use a z/OS-based debugging engine to debug code in an MVS project, which must be in online state; or
- You use a workstation-based debugging engine to debug code in a local project.

For either local code or for z/OS-based code the standard functionality of a debugger is available. On z/OS, you can debug load modules that run on MVS, including load modules that run in a subsystem (CICS or IMS). Debugging can occur only if a debug daemon is listening. You can debug MVS batch or TSO load modules and interactive CICS or IMS programs.
Editors
z/OS application development tools allow you to use any text editor and also provides two:

- The z/OS Live Parsing Extensible Editor (LPEX), which is integrated into the Workbench
- The Program editor, which is launched outside of the workbench

It is recommended that you use the z/OS LPEX Editor, which is particularly useful as you develop COBOL, as:

- You can edit remote z/OS source files, while maintaining z/OS attributes such as record length and sequence numbers.
- Remote files you are editing are locked by way of ISPF ENQ/DEQ, so that other users (ISPF or z/OS LPEX) cannot edit them at the same time.
- You can save keystrokes by invoking code-assist to complete variable names, COBOL statements, and EXEC CICS statements.
- You can quickly open a file referenced in a COPY statement by double-clicking the file name.

Both editors highlight various aspects of the language-specific syntax which includes COBOL, PL/I, HLASM, and JCL. Both editors can be customized.

7.18.2 Required z/OS server tools

Included with the Enterprise Developer are some required tools that must be installed on z/OS to support the development and execution of programs created in the workbench.

WebSphere Studio Enterprise Developer options for z/OS
These server components are necessary for COBOL and PL/I program development with Enterprise Developer:

- A 390 build server used for building native COBOL and PL/I programs
- The modules necessary for the COBOL and PL/I tools to receive error feedback when doing remote project builds

IBM OS/390 Foreign File System server
The foreign file system (FFS) server and the FFS client installed on the workstation work together to provide Enterprise Developer users with transparent access to their MVS data sets plus other components for monitoring jobs and submitting TSO commands.
Installed server components include:

- The foreign file server, which enables remote edit, compile, and debug of COBOL or PL/I.
- The Job Monitor Server, which enables the job monitoring function.
- The TSO Command Server, which enables the ability to issue TSO commands from Enterprise Developer.
- PTF IP22615 for IBM OS/390 Foreign File Server, which provides the updates that enable it to work with Enterprise Developer.
- REXX/370 Alternate Library. This provides the code necessary to execute the SCLM interface provided in WSED Options for z/OS if you do not have the REXX/370 Runtime library already installed on the z/OS system.

### 7.19 XML enablement (host) tools

XML Enablement for the Enterprise is a tool that lets you easily adapt COBOL-based business applications so that they can process and produce XML messages. The tool's primary use is to provide a new kind of access to a called application, so that (for example) an Internet user can access an existing CICS application. The tool also can help you to embed a COBOL application in a larger system that uses XML for data interchange.

In the case of a called COBOL program that you want to enable, you leave the called application as is, and other legacy programs that supply COBOL data (rather than XML) can access the COBOL program as before. Restrictions apply, however, as described later.

#### 7.19.1 Inputs

The input to XML Enablement for the Enterprise is either a COBOL source program or a COBOL COPY file. The COBOL source remains unchanged.

#### 7.19.2 Outputs

The tool generates the following outputs:

- Two COBOL-based converter programs: One to transform XML input messages into native COBOL data, one to transform COBOL data into an XML output message.
- A COBOL-based driver program that shows how to invoke these programs in order:
  a. The input converter
Chapter 7. WebSphere Studio V5 overview and architecture

7.19.3 Restrictions

Restrictions are as follows:

- Each XML schema definition must be based on a single 01-level data declaration. (The definition for input and output may be the same.)
- The data declarations must not include objects, pointers, or procedure pointers.
- If a called COBOL program receives data by way of a user interface (by displaying a CICS map, for example), you must take the user interaction out of that program; the called COBOL must be invoked by a program rather than by a user interface.
- If you want a COBOL program to send an XML message, you need to change the logic of that program:
  - The caller must invoke an output converter to create an XML message from the COBOL data that is required by the called program.
  - The caller then includes the resulting XML string (not the COBOL data) in the call to the called program.

7.20 WebSphere Studio Client for iSeries tools

The WebSphere Development Studio Client for iSeries V5.0 product includes either WebSphere Studio Site Developer or WebSphere Studio Application
Developer, depending on the edition purchased, and installs iSeries-specific tools into the WebSphere Studio programming environment.

It also includes the IBM Distributed Debugger, as well as the CoOperative Development Environment (CODE) and VisualAge RPG, both of which can be launched from the workbench.

This section introduces the primary tools that are added as plug-ins to WebSphere Studio.

### 7.20.1 WebFacing tool

The IBM WebFacing tool is used to convert iSeries host applications into Web applications and consists of four different components:

- A WebFacing perspective showing all necessary objects to work with the IBM WebFacing tool.

- The WebFacing Tool Project wizard allowing you to enter the necessary information needed to build a WebFacing application and storing that information as a WebFacing project in your workspace.

- The WebFacing Conversion wizard, which reads the information in the WebFacing project, supplied by you either through entering into the WebFacing Tool Project wizard or by changing certain properties at a later time.

  The wizard then converts each record format defined in the DDS members to JSPs, JavaScript, Java beans, XML files, and HTML files. All of these artifacts together form a Web application ready to be deployed to an application server such as the WebSphere Application Server Version 3.5.5 or later.

  You can enhance and modify the source code by adding buttons, changing styles, and enhancing layouts (for example). When the project is converted again, the code is overwritten with the new generated source code.

- The WebFacing runtime server, which is part of OS/400 V4R5 or later and allows the WebFacing application to start and connect to the existing 5250 application.

  You can use Remote System Explorer (RSE) to transfer WAR and EAR files to an iSeries host.

### 7.20.2 Web tools - iSeries extensions

These tools are designed specifically for use with iSeries.
Web Interaction wizard
The Web Interaction wizard generates a Web interaction, which connects to an existing OS/400 program object or an Integrated Language Environment (ILE) procedure. Such an interaction accepts parameters from an input JSP; passes them to the OS/400 program through iSeries runtime classes, which are also generated by the wizard; and then sends the output of the program to an output JSP. Finally, it generates a Struts action class that ties all the pieces together.

An interaction is defined by the communication that occurs between Web pages and the business processes. In addition, you can use the wizard to define message handling, session, flow-control, and other configuration options for the Web pages in the application.

The wizard uses information entered by the application developer into a series of dialogs to add code to the JSP files, update the Struts configuration file, and generate the iSeries classes that communicate between the Web pages and traditional OS/400 programs or Java beans that execute the business logic for an application.

The input and output JSP files can be created manually before using the Web Interaction wizard or the wizard can generate the input page, the output page, or both.

iSeries Web Tools Runtime Configuration wizard
This wizard complements the Web Interaction wizard. It is used to specify, per Web project, runtime information that all wizard-generated interactions use. This information includes sign-on information and library list setup.

iSeries Web components
iSeries Web page widgets are available in the Page Designer tool. They can be inserted into a Web page, and generate HTML and JavaScript on your behalf, based on properties you specify.

7.20.3 Java tools for iSeries
WebSphere Studio Java Development Tools (JDTs) along with iSeries Java development tools, give you the ability to develop Java applications and write,
compile, test, debug, and edit programs written in the Java programming language. Specific iSeries Java tools are:

- Program Call wizard that generates a Java bean containing the Java code needed to call iSeries programs. You can also use the output to create Web services that call iSeries programs.
- The File Export/Import wizard allows you to import and export files to and from the integrated files system on an iSeries server into a Java project.
- The iSeries Java Transform and Run view, for remotely transforming, running, and debugging Java applications.
- Pre-supplied Java beans: Data File Utility bean, Swing JFormatted bean, Object List bean, and a Database Field List bean.
- Java Tools for iSeries examples.

### 7.20.4 Remote System Explorer (RSE) perspective

The Remote System Explorer is a workbench perspective that provides you with access to all the objects on an iSeries system. It can be thought of as an enhanced and more flexible workstation version of the Programming Development Manager (PDM), allowing you to effectively manage and organize iSeries resources and applications.

Figure 7-8 shows the RSE interface after a connection to a iSeries system has been made.

![Figure 7-8 Remote System Explorer for iSeries](image)
The RSE displays four subsystems:

► iSeries objects: You can create, copy, rename, delete, and retrieve information about objects. You can also run programs and compile source members.

► iSeries commands: You can run pre-defined commands and command sets or create and run your own.

► iSeries jobs: You can monitor and manage jobs running on an iSeries host.

► Integrated files system (IFS): The IFS provides a common interface to the different file systems that are available on an iSeries host.

**Edit source members**
From the RSE you can navigate to a source member in the Remote Systems view and with a simple double-click launch the Live Parsing Extensible Editor (LPEX Editor) to edit the source code.

**Compile source members**
Likewise, you can compile a source member from the Remote Systems view. Selecting a source member and right-clicking brings up a contextual menu with compile choices corresponding to the type of source member selected. Activating a menu choice launches an interface where you can choose compile options and command parameters.

### 7.20.5 iSeries Projects perspective
The primary focus of the iSeries Projects perspective is on disconnected iSeries development on a Windows platform. The intent is that you either create iSeries resources locally or import existing iSeries resources, then edit the resources locally in an iSeries project. When you have finished updating your files and members locally, you upload these resources to the iSeries host and submit a build of your application.
Recognizing the e-commerce needs of small, medium, and large companies, IBM has developed several editions of IBM WebSphere Commerce V5.5, including Express Edition, Professional Edition, and Business Edition. Each of the product editions includes a secure and scalable runtime architecture, a flexible and robust J2EE programming model, and sample stores to speed application development.

This redpaper provides an architectural overview of the IBM WebSphere Commerce V5.5 runtime architecture, business and store models, and programming model.

The redpaper is organized into the following topics:

- Runtime architecture
- Business and store models
- Programming model

**Note:** For more detailed information, refer to the *WebSphere Commerce V5.5 Handbook, Customization and Deployment Guide*, SG24-6969.
8.1 Runtime architecture

This section identifies the key components of the WebSphere Commerce runtime architecture. We have categorized the components for the WebSphere Commerce Server as follows (see Figure 8-1 on page 3):

- **WebSphere Commerce software components**
  
  We have listed the primary software components for WebSphere Commerce. Note that there are many additional software components that can be integrated with WebSphere Commerce that are included in the IBM WebSphere Commerce V5.5 product packaging that have not been listed.
  
  - Web server
  - WebSphere Application Server
  - Database server
  - WebSphere Commerce Server
  - WebSphere Commerce Payments Server
  - Enablement software (optional software components)

- **WebSphere Commerce Server subsystems**
  
  The subsystems run within the WebSphere Commerce enterprise application on the WebSphere Commerce Server, and provide the infrastructure to support the features used by the administration tooling and stores.
  
  - Member subsystem
  - Catalog subsystem
  - Trading subsystem
  - Order subsystem
  - Merchandizing subsystem
  - Marketing subsystem
  - Inventory subsystem
  - Messaging subsystem

- **Common server runtime (framework)**
  
  The common server runtime provides a framework in which the commerce applications are deployed and executed.

- **Business Interaction Engine**
  
  The subsystems and server runtime operate within an interaction engine that provides all of the components with the necessary business context, and governed by the contextual frameworks such as policies, entitlement, stores, personalization, and globalization.

- **Administration tools**
  
  The administration tools are used to configure and manage the WebSphere Commerce site and store operations:
- WebSphere Commerce Configuration Manager
- WebSphere Commerce Administration Console
- WebSphere Commerce Accelerator
- WebSphere Commerce Organization Administration Console
- WebSphere Commerce Payments Administration Console
- WebSphere Commerce Scheduler
- WebSphere Commerce Loader Package

Figure 8-1 WebSphere Commerce Server runtime components
8.1.1 WebSphere Commerce software components

There are many software components included with IBM WebSphere Commerce V5.5. In this section we will limit our discussion to the following software components of the WebSphere Commerce runtime architecture, depicted in Figure 8-1 on page 3:

- Web server
- WebSphere Application Server
- Database server
- WebSphere Commerce Server
- WebSphere Commerce Payments Server

Note: The WebSphere Commerce node architecture depicted in Figure 8-1 on page 3 shows a single-tier configuration. The software components listed in this section can be distributed on separate nodes for scalability and security reasons.

For more detailed information on runtime configurations refer to the WebSphere Commerce V5.5 Handbook, Customization and Deployment Guide, SG24-6969, the “Runtime topology selection” chapter.

Web server

The Web server can be installed on the WebSphere Commerce node or a remote node, which can be optionally clustered for load balancing using the Edge Components of WebSphere Application Server. Regardless of whether the Web server is local or remote, it must be configured to use the WebSphere Application Server plug-in. There are several supported Web server plug-ins. The IBM HTTP Server and plug-in are found on the WebSphere Application Server CD.

The majority of the WebSphere Commerce tooling and store application assets are J2EE components (JSPs, servlets, EJBs, etc.) and run on the application server located on the WebSphere Commerce node. There are some static HTML pages and images found in the WebSphere Commerce tools and stores that can be served by the Web server.

Incoming HTTP requests from Web browser clients are received by the Web server and WebSphere plug-in. The WebSphere plug-in, via the use of the plugin-cfg.xml file, redirects requests to applications on the WebSphere Application Server on the WebSphere Commerce node.

In the event that you want to set up a remote Web server, you must manually configure the WebSphere Application Server virtual host aliases and regenerate the WebSphere plugin-cfg.xml file on the WebSphere Commerce node. The updated plugin-cfg.xml file must then be manually copied to the remote Web server. The Web server hostname must be updated in the WebSphere
Commerce instance and WebSphere Commerce Payments instance via the
Configuration Manager. In addition, static application assets that are to be served
by the Web server (HTML, images), must also be manually copied to the remote
Web server. Once the remote Web server has reloaded the plugin-cfg.xml file
and the WebSphere Application Server has been restarted to reflect the
configuration changes. The remote Web server hostname can now be used by
the Web browser client to access the WebSphere Commerce tools and stores.

**WebSphere Application Server**
The WebSphere Commerce Server leverages the J2EE technologies provided by
the WebSphere Application Server such as JSPs, servlets (WebSphere
Commerce commands), EJBs, XML, Web Services, security, MQ embedded
messaging, etc.

IBM WebSphere Commerce V5.5 includes two WebSphere Application Server
Network Deployment. Network Deployment is suitable for single-tier, two-tier, and
three-tier runtime configurations. When multiple WebSphere Application Servers
are needed for scalability such as horizontal application clustering, then the
Network Deployment Edition is needed.

**Database server**
IBM DB2 Universal Database V8.1, Enterprise Server Edition is included with
WebSphere Commerce V5.5. In addition, Oracle9i (9.2.0.1) Enterprise Server
and Standard Edition are supported (not included). The database server is used
for the WebSphere Commerce instance database and the WebSphere
Commerce Payments database.

The WebSphere Commerce instance database is used to store configuration
data such as taxes, shipping and customer profile information, and the product
catalog. There are over 600 tables and numerous columns built-in to the
WebSphere Commerce V5.5 schema.

The WebSphere Commerce Payments database is used for payment
configuration such as accounts, payment types, cassettes, and payment
transaction data.

The database server software can be installed on the same node as WebSphere
Commerce or on a remote database server node.

**WebSphere Commerce Server**
The WebSphere Commerce Server is a WebSphere enterprise application,
which runs on its own application server within the WebSphere Application
Server. The WebSphere Commerce application software is installed via the
WebSphere Commerce Installer.
After installation, WebSphere Commerce must be configured using the Configuration Manager. The Configuration Manager is used to create the WebSphere Commerce instance. During instance creation an application server for the WebSphere Commerce Server and the enterprise application is deployed.

For most runtime topologies, the configuration of the WebSphere Application Server is performed for the user via the WebSphere Commerce Configuration Manager. When adding a remote Web server, remote WebSphere Commerce Payments node, and clustering using WebSphere Application Server Network Deployment, some manual configuration is needed.

**WebSphere Commerce Payments Server**

The WebSphere Commerce Payments Server is a WebSphere enterprise application, which runs on its own application server within the WebSphere Application Server.

After installation, the WebSphere Commerce Payments must be created using the Configuration Manager. When creating a WebSphere Commerce Payments instance using the Configuration Manager, the WebSphere Commerce Payments instance application server is created and the WebSphere Commerce Payments enterprise application is deployed.

The WebSphere Commerce Payments Server can be installed on the WebSphere Commerce node or on a remote node.

**Enablement software**

There are several enablement software components included with WebSphere Commerce that can be optionally installed. In addition, we have listed WebSphere enablement software that can be leveraged by WebSphere Commerce.

**WebSphere Commerce enablement software**

This section describes the functionality provided by the following enablement software included with IBM WebSphere Commerce V5.5:

- **Personalization**
  
  There are two personalization solutions included with WebSphere Commerce: Blaze Rules and Likeminds collaborative filtering. The personalization software can be used to improve the customer experience by tailoring the site to a number of criteria such as customer profile, shopping cart contents, and purchase history.

- **Commerce analytics**
  
  WebSphere Commerce V5.5 has improved analytics capability through the use of WebSphere Commerce Analyzer and IBM DB2 Intelligent Miner for
Data V8.1 and IBM DB2 UDB V8.1 warehousing features. There are many business intelligence reports included with WebSphere Commerce, which leverage the technology of the WebSphere Commerce Reporting Framework and are accessible from the WebSphere Commerce Accelerator.

- Messaging integration
  WebSphere Commerce V5.5 includes messaging adapters for HTTP, e-mail, MQ, InterChange Server (ICS), and file. Inbound messaging supports the HTTP and MQ adapters and can be customized to support other protocols. Outbound messaging support includes WebSphere MQ, WebSphere InterChange Server (ICS), e-mail, and file adapters. WebSphere Commerce ships with a sample J2C-based connector that can be customized.

- Collaboration
  Customer Care, which is a Lotus Sametime V3.0 application integrated with WebSphere Commerce, provides live help in real time between customers and customer service representatives (CSRs).

  Collaborative Workspaces, which is built upon Lotus QuickPlace V3.0 and only available with the Business Edition, provides an environment where business customers and line-of-business users can interact.

- Directory Server (LDAP)
  IBM Directory Server V4.1.1 is an LDAP Directory Server included with WebSphere Commerce V5.5. WebSphere Commerce and can optionally be configured to use LDAP as the member repository instead of the default WebSphere Commerce instance database. LDAP provides for better integration and single sign-on (SSO) for multiple participating applications sharing the same LDAP directory.

**WebSphere enablement software**

The IBM WebSphere software brand includes many enablement software solutions that can be leveraged by WebSphere Commerce. This section highlights the WebSphere Foundations, Tools, and Business Portals.

- WebSphere Foundation and Tools
  The WebSphere Application Server is a J2EE-based application server. The WebSphere Commerce Server (application server) is an enterprise application and has its own application server. WebSphere Commerce V5.5 includes WebSphere Application Server and WebSphere Application Server Network Deployment. WebSphere Commerce can leverage the technologies provided by the WebSphere Application Server and WebSphere tooling, namely WebSphere Studio Application Developer for the development of Java and related application assets.
WebSphere Business Portals

WebSphere Business Portals help extend and personalize the user experience. The integration of WebSphere Commerce V5.5 and WebSphere Portal V5 will be delivered in an upcoming Enhancement Pack.

8.1.2 WebSphere Commerce Server subsystems

The subsystems provide a great deal of functionality for the WebSphere Commerce Server. The subsystems are used to integrate other components and external nodes as well as make it vastly easier for the store developer to customize, deploy, and manage a store.

Member subsystem

The Member subsystem is a component of the WebSphere Commerce Server that provides a framework for managing the following participants of the system:

- Organizational entity (for example, IBM)
- Organizational unit (for example, IBM Software Group)
- Member group - Group of users
- Members - Users, member groups, organizational entity

The member data is stored within either a WebSphere Commerce instance database or an LDAP directory server database. By default, WebSphere Commerce uses the instance database as its registry.

The major functions of the Member subsystems are to provide member registration and profile management. Other closely related services of the Member subsystem include authentication, access control, and session management.

User registration methods

To facilitate various requirements for e-commerce Web sites, WebSphere Commerce provides several methods for user registration, as seen in Figure 8-2 on page 9.
Below we describe the steps shown in Figure 8-2.

1. **WebSphere Commerce online registration**

   This involves the registration of members online for the e-commerce Web site. Users will be prompted for registration before catalog navigation or during the order checkout process. This is the most common and direct approach of the user registration method. This method does not support mass registrations.

2. **MQ**

   WebSphere Commerce also supports member registration from back-end systems, such as ERP systems using WebSphere MQ. To enable this method, the WebSphere Commerce message transport adapter and MQ need to be configured. WebSphere Commerce provides an inbound messaging service for creating and updating customer registration. This method is very useful if you have legacy systems, which you need to Web enable. This approach is best suited for an enterprise integration solution. By default, WebSphere Commerce does not provide outbound services over MQ for the Member subsystem.

3. **Using LDAP**

   WebSphere Commerce also supports integration with an industry-standard LDAP directory for user registration. If LDAP is used as a user registry, then WebSphere Commerce will synchronize with the LDAP directory, based on the mapping parameters defined in the WebSphere Commerce ldapentry.xml file between WebSphere Commerce and LDAP. When the registered user in
LDAP logs into the WebSphere Commerce system, the user entry is replicated on the fly to the WebSphere Commerce store database. WebSphere Commerce will synchronize with the LDAP directory to retrieve and update the user registration.

4. Using WebSphere Commerce Loader Package (MassLoad)

WebSphere Commerce Loader Package is used for mass database updates. The MassLoad tool can be used for mass registration of users. This method is especially useful in the migration from previous versions, in database management, and in member registration exchange across WebSphere Commerce systems.

The registered users will manage their user profiles by updating the registration information, and adding, modifying, or deleting address entries in the address book. Also, a customer service representative can update the user profiles.

**Member security services**

The following security services are closely related to the Member subsystem:

- **Roles**
  The Member subsystem allows its users and organizational entity members to be assigned roles depending on the activities in which they are allowed to perform. Role assignment is the responsibility of the site administrator.

- **Authentication**
  WebSphere Commerce supports two modes of authentication:
  - Basic authentication (using user ID and password)
    This mode of authentication is the default and can be used with the WebSphere Commerce store database or an LDAP directory.
  - Certificate-based authentication (using x.509 certificates)
    The authentication mode is configured via the WebSphere Commerce Configuration Manager within the Web server tab of the instance properties.

- **Access control**
  To facilitate database management and ensure security, access to WebSphere Commerce must be restricted to specific individuals and organizations. The process of restricting access is referred to as access control. Access control can be defined as security guidelines that:
  - Allow or deny a user of a system access to the resources managed by a system.
  - Specify what actions the user can perform on each resource.
Access control is managed through the implementation of access control policies and policy groups.

- Access control policies

An access control policy authorizes a group of users to perform particular actions on a group of WebSphere Commerce resources. Unless authorized through one or more access control policies, users have no access to any functions. Access control policies grant authorization to a specific group of users to perform particular actions on resources in a specified resource group.

An access control policy consists of four parts:

- Access group: The group of users to which the policy applies.
- Action group: A group of actions.
- Resource group: The resources controlled by the policy. A resource group may include business objects such as contract or order, or a set of related commands.
- Relationship (optional): Each resource type can have a set of relationships associated with it. Each resource can have a set of users that fulfill each relationship.

- Policy groups

Different organizations in an e-commerce site require different sets of access control policies. For example, a seller organization would require shopping-related policies, while a buyer organization would not need them. In order to accomplish this type of requirement, in WebSphere Commerce, access control policies are partitioned into access control policy groups. In order for an access control policy to be applied in the site, it must belong to an access control policy group. Then, based on their business and access control requirements, organizations subscribe to the appropriate access control policy groups.

- Session control

WebSphere Commerce is a WebSphere application that is based on the J2EE specification. For this reason, WebSphere Commerce follows the servlet specification for session management.

- Session Manager: You can configure WebSphere Commerce session manager from the Session Management tab via the Configuration Manager to use either WebSphere Commerce or WebSphere Application Server.

The WebSphere Commerce session manager offers better performance, but does not allow extra information to be added to the session, which the WebSphere Application Server does.
Session types: WebSphere Commerce supports two types of session management: Cookie based and URL rewriting. For security reasons, cookie-based session management uses two types of cookies:
Non-secure session cookies, which are used to manage the session data;
and secure authentication cookies used to manage authentication data.

**Single sign-on**
WebSphere Commerce supports single sign-on when configured with an LDAP directory.

**Catalog subsystem**
The Catalog subsystem provides online catalog navigation, merchandising features, interest lists, and search capabilities. The Catalog subsystem includes all logic and data relevant to a catalog, including categories, products and their attributes, items, and any associations or relationships among them. It interacts with the Member subsystems and the Order subsystems to obtain information about viewing templates and pricing. The following features are provided:

- **Groupings**
  A generic grouping construct is introduced for categorizations of various products. The owner of a catalog group may not necessarily be the owner of all the catalog entries in the group. This allows portal owners to define the categories of products offered while other suppliers can add their products to the catalog group.

- **Catalog entries**
  One or more catalog entries can belong to a catalog group. A set of base object types is provided to represent products, stock keeping unit (SKU) items, packages, and bundles in a catalog entry.

- **Merchandising associations**
  These make it possible to create an association between any two catalog objects, which become cross-sells, up-sells, and promotions.

- **Globalization support**
  The catalog design addresses the requirement to support multicultural features such as product display and currency format according to the locale.
Trading subsystem
The Trading subsystem in WebSphere Commerce provides the logic, function, and data relevant for negotiating the price and quantity of a product or set of products between the buyer and seller organization. For the Professional Edition the trading subsystem includes auctions. For the Business Edition the trading subsystem includes auctions, contracts, and Request for Quote (RFQ) components that are used to carry out specific transactions between organizations.

Order subsystem
The Order subsystem is a component of the WebSphere Commerce Server that provides shopping carts, order processing, and order management function support. Related services, such as pricing, taxation, payment, inventory, and fulfillment, are also part of the order subsystem. Order processing capabilities include quick order or buy, scheduled orders, multiple pending orders, reorders, and splitting or back orders.

Merchandizing subsystem
The Merchandizing subsystem is a component of the WebSphere Commerce Server which provides functionality for cross-selling, up-selling, suggested accessories, and merchandizing associations between products in the catalog.

Marketing subsystem
The Marketing subsystem is a component of the WebSphere Commerce Server, and provides numerous marketing concepts to your site. Components of the marketing subsystem provide functionality to create marketing campaigns including product recommendations, advertisements, and electronic coupons; discounts; customer profiles; and collaboration.

Inventory subsystem
The Inventory subsystem provides real-time inventory management. Components of the inventory subsystem provide functionality to record inventory received from vendors and that is returned by customers, adjust inventory quantity, determine the disposition of returned inventory, and ship and receive inventory.

Messaging subsystem
The Messaging system provides WebSphere Commerce with the ability to communicate externally. This communication includes sending messages to and receiving messages from back-end systems or external systems, as well as sending notification to customers and administrators that events have occurred within WebSphere Commerce. This is accomplished through two subsystems.
An inbound system that manages inbound messages coming from back-end and external systems, and an outbound messaging system that allows you to send notification to users as well as outbound messages to back-end systems and external systems.

For example, you can set up the messaging system to send e-mail messages notifying your customers that their orders have been shipped. The messaging system provides a mechanism for integrating WebSphere Commerce with back-end systems. You can configure WebSphere Commerce to send an outbound message to a back-end system whenever an order is created at your store. This order information can be used by the back-end system to do necessary order fulfillment processing. The back-end system can later send order status messages back to WebSphere Commerce indicating that order delivery has taken place or an order invoice has been issued. E-mail can also be sent to update the customer.

8.1.3 WebSphere Commerce administration tools

IBM WebSphere Commerce V5.5 and supporting software includes many administration tools that make managing the WebSphere Commerce site and store easier for IT specialists and line-of-business (LOB) users. All of the administration tools listed in this section are included with the IBM WebSphere Commerce V5.5 product packaging (Business and Professional Edition).

As seen in Figure 8-1 on page 3, WebSphere Commerce V5.5 includes a set of Web browser based administration tools, and the stand alone Java-based configuration tool (Configuration Manager). This section highlights the functionality provided by each of the WebSphere Commerce administration tools.

**WebSphere Commerce Configuration Manager**

After installing WebSphere Commerce the WebSphere Commerce node must be configured. The WebSphere Commerce Configuration Manager is used to create, modify, and delete various components of a WebSphere Commerce instance and the WebSphere Commerce Payments instance, which is new to this release of WebSphere Commerce.

Some of the key tasks performed when creating a WebSphere Commerce instance include the following:

- Create the WebSphere Commerce application sever.
- Deploy the WebSphere Commerce enterprise application.
- Update the WebSphere Commerce `<instance_name>`.xml file with the settings specified during instance creation.
Create the WebSphere Commerce instance database and populate the database with the WebSphere Commerce schema.

Configure the Web server configuration file with virtual hosts, aliases, and directives.

The Configuration Manager provides a GUI interface to the WebSphere Commerce <instance_name>.xml configuration file.

**Note:** Whenever possible, components should be enabled through the Configuration Manager rather than by modifying the WebSphere Commerce <instance_name>.xml configuration file.

The Configuration Manager consists of a Java application based server and client. The Configuration Manager Server must be started first. This is accomplished slightly differently on Windows, AIX, and Solaris platforms. On Windows, the Configuration Manager Server started as a Windows services (IBM WC Configuration Manager). When the Configuration Manager (client) is started from the WebSphere Commerce application folder, the user is prompted for a password. By default the user ID is webadmin, and the password is webibm. During the first logon, you will be prompted to enter a new password.

**WebSphere Commerce Administration Console**

The WebSphere Commerce Administration Console allows you to control your site or store by completing administrative operations and configuration tasks. If you are a site administrator, you select the store and language with which you want to work when you log on to the Administration Console. The tasks that you are authorized to perform display on the Administration Console home page through various menus. These tasks are based on the user group names (roles) and authority levels.

The WebSphere Commerce Administration Console is installed by default as part of the WebSphere Commerce installation, but not deployed until the WebSphere Commerce instance has been created. The WebSphere Commerce Administration Console is accessible to Web browser clients (Microsoft Internet Explorer 5.5 or later) by entering the following URL:

```
https://<wc_hostname>:8002/adminconsole
```

There are two things to take note of with the Administration Console URL. First, the Administration Console is configured to use SSL and requires the HTTPS protocol. Second, the 8002 port is defined when creating the WebSphere Commerce instance. The port 8002 is added to the Web server configuration file (for example, httpd.conf) under a virtual host for this server, as well as the alias /adminconsole.
After logging on to the WebSphere Commerce Administration Console as the site administrator (user defined during instance creation), you will be prompted to select site or store to configure. The site option will provide a super set of menu options from the store option.

The WebSphere Commerce Administration Console from the site menus includes functionality for the following site level options:

- **Security**
  - Account Policy
  - Password Policy
  - Account Lockout Policy
  - Security Checker

- **Configuration**
  - Transports (e-mail, WebSphere MQ, WebSphere InterChange Server, file)
  - Message Types
  - Component Configuration
  - Scheduler
  - Registry
  - View Unsent Message
  - View Archived Messages
  - Product Information

- **Payments**
  - User
  - Merchant Settings
  - Payment Settings
  - Cassettes

- **Store Archives**
  - Publish
  - Publish Status

**Note:** In previous releases of the WebSphere Commerce (V5.1, 5.4) store archives were published from Store Services. The store publishing functionality has been moved to the Administration Console under the Store Archives menu option.

After logging in to the Administration Console, if you selected store you will see a listing of stores that have been published. The WebSphere Commerce
Administration Console from the store menus includes functionality for the following store level options:

- Access Management
  - Policies
- Configuration
  - Transports
  - Message Types
  - E-mail activities
  - Scheduler
  - View Unsent Message
  - View Archived Messages
- Rules Service
  - Administration
- Payments
  - Users
  - Merchant Settings

**WebSphere Commerce Accelerator**

The WebSphere Commerce Accelerator allows you to maintain online stores, hubs, and catalogs by completing various store operations, from managing the look and feel of your store to creating and maintaining orders to tracking store activities.

If you are authorized to work with multiple stores, when you log onto the WebSphere Commerce Accelerator, you select the store and language with which you want to work. If you are authorized to work with a single store, the store name is pre-selected during logon. Additionally, if the store supports more than one language, you can select the language with which you want to work. Finally, if you are assigned a role with fulfillment duties, you can also choose the fulfillment center associated with the store when you log on. If you wish to change your store, language, or fulfillment center selection, click the icon, found in the upper left corner, to display the selection dialog.

Tasks that you are authorized to perform in your role are displayed on the WebSphere Commerce Accelerator home page menus. These tasks are based on user roles, authority levels, and the business model and type of store. If you need to change your access level, contact your site administrator. To return to the WebSphere Commerce Accelerator home page, at any time, click **Home** near the top of the WebSphere Commerce Accelerator.
The WebSphere Commerce Accelerator is installed by default as part of the WebSphere Commerce installation, but not deployed until the WebSphere Commerce instance has been created. The WebSphere Commerce Accelerator is accessible to Web browser clients (Microsoft Internet Explorer 5.5 or later) by entering the following URL:

https://<wc_hostname>:8000/accelerator

The WebSphere Commerce Accelerator includes the following menu options:

- **Store**
  - Open/Close
  - Change Profile
  - Change Flow
  - Change Shipping
  - Change Tax
  - Payment Settings
  - Approval Requests
  - Find Approval Requests
  - Approval Submissions
  - Find Approval Submissions
  - Fulfill Centers
  - Return Reasons
  - Report Delivery Status
  - Business Intelligence Reports
  - Collaborative Workspaces

- **Sales**
  - Accounts
  - RFQs
  - Personalized Attributes
  - Find Customers
  - Find Orders
  - Find Returns
  - Order Management Reports
  - Operational Reports
  - Auctions
  - Customer Care
  - Customer Care Queue
  - Approve Payment
  - Deposit Payment
  - Settle Payment
  - Find Payment
  - Find Payment Batch
WebSphere Commerce Organization Administration Console

The Organization Administration Console allows the buyer administrators and you to control the organizations that access your site or store. Tasks that you are authorized to perform in your role are displayed on the Organization Administration Console home page menus. These tasks are based on user roles...
and authority levels, which are defined in XML files on your WebSphere Commerce system and are assigned by the site administrator by using the Administration Console.

The WebSphere Commerce Organization Administration Console is installed by default as part of the WebSphere Commerce installation, but not deployed until the WebSphere Commerce instance has been created. The WebSphere Commerce Organization Administration Console is accessible to Web browser clients (Microsoft Internet Explorer 5.5 or later) by entering the following URL:

https://<wc_hostname>:8004/orgadminconsole

After logging onto the WebSphere Commerce Organization Administration Console you will see the following menu options:

- **Access Management**
  - Users
  - Organizations
  - Roles
  - Access Groups
  - Policies
  - Resource Groups
  - Action Groups

- **Approvals**
  - Approval Requests
  - Find Approval Requests

**WebSphere Commerce Payments Administration Console**

In WebSphere Commerce V5.5, payment administration can be performed from the WebSphere Commerce Administration Console (configuration) and the WebSphere Commerce Accelerator (operations). The WebSphere Commerce Payments Console is a legacy component of the WebSphere Commerce Payments available in previous releases.

The WebSphere Commerce Payments Console is accessible to Web browser clients (Microsoft Internet Explorer 5.5 or later) by entering the following URL:

https://<hostname>:5433/webapp/PaymentManager

The hostname is the Web server configured for the WebSphere Commerce Payments virtual host.

We find using the WebSphere Commerce Payments Console useful for verifying that the WebSphere Commerce Payments instance was created properly and for approving the payment of a transaction for testing purposes.
WebSphere Commerce Scheduler
The WebSphere Commerce Scheduler is a background server that schedules and launches jobs, both at the site and store levels. From the Configuration menu of the Administration Console, the scheduler allows site administrators to schedule and configure jobs.

The scheduler polls the SCHACTIVE table to find jobs scheduled to run. The following list describes the possible entries for the STATE column:

- W indicates that the job is waiting to be executed if necessary.
- I indicates that the job is currently inactive.
- IF indicates that the job has run and failed and the instance will rerun the job.

For each entry where the state is W and the preferred start time is less than or equal to the current time, the scheduler gets the job's configuration information from the SCHCONFIG table. If the INTERFACENAME field is defined, the scheduler gets the implementation of the business logic task command and if that implementation uses the business logic interface, the scheduler runs the business logic task command. If no exceptions are thrown, the state is changed to I.

When the scheduler finds entries in the SCHACTIVE table where the state is I or IF and the preferred start time is equal to or less than the current time, it executes the job.

The scheduler is scalable across one or more threads, running on one or more machines. Multiple commerce servers or clones can connect to the same database. When a job is added to the SCHCONFIG table, the job can be scheduled to run on any WebSphere Commerce Server or application server cluster.

WebSphere Commerce Loader Package
WebSphere Commerce V5.5 has two catalog management tools, Loader Package and the Accelerator Product Management Tool. The Accelerator Product Management Tool is GUI based and very well suited for operations when only a few product entries need to be performed. The WebSphere Commerce Loader Package is a Java-based set of utilities that can be executed via the command line or via a Java API and allows for automation and is more appropriate for managing large product catalogs.

Note: The Loader Package was known as Catalog Manager in WebSphere Commerce V5.4.
The Loader Package provides the following functionality:

- Aggregate data from multiple input streams into one aggregated database.
- Transform data from ASCII-based CSV to XML format. The MassExtract tool will extract data, but to XML format. Converting from XML to CSV is not as easy as converting CSV to XML.
- Remap data from one XML format to another.
- Import data from multiple input sources in the form of ASCII-based CSV and XML files into WebSphere Commerce.

The WebSphere Commerce Loader Package includes the following components:

- **Text Transform tool**
  The Loader Package Text Transform tool provides the ability to convert ASCII-file output such as a CSV file into an XML data format that later can be transformed and eventually loaded into the WebSphere Commerce database.

- **XSL Editor**
  A major benefit of using XML for data representation and structuring is that many tools exist to help with transformation of XML data from one format to another. An important method involves the use of XML style sheets, which are written in the eXtensible Stylesheet Language, or XSL. These style sheets may then be applied to XML documents by a tool that conforms to the XSL Transformation standard, or XSLT.
  The WebSphere Commerce Catalog Manager XSL Editor is used to develop XSL style sheets.

- **XML Transform tool**
  This Loader Package tool takes XSL style sheets, such as those created using the XSL Editor, and applies the transformation to XML files, such as those created by the Text Transform tool. The result is normally another XML file that is structured according to the input data requirements for a target application such as WebSphere Commerce.

- **DTD Generator**
  The DTD Generator can create a DTD and a schema to use with the Loader Package. The DTD Generator uses an input file containing database-table names and generates either a DTD or a DTD and an XML schema to describe the selected database tables.

- **ID Resolver**
  The ID Resolver tool takes an XML file and compares the contents to data within a specified database. Each element within the XML file is examined in turn, and if it contains attributes that require resolving, the data from other
attributes within the element is used to find a match with data already within
the database, or that has been previously specified within the XML file.

- **Loader**

  After ID resolution, the loading of the newly resolved data into the database is
  performed using the loader, known as the MassLoad tool. This Java
  application effectively takes a fully resolved XML file and loads each element
  in turn into the database. This is performed using the MassLoad tool.

  An essential prerequisite before attempting to load data into the commerce
database is that it *must* have been successfully and completely resolved.

- **Extractor**

  To extract data from a database using the Extractor, you must specify the data
  that you want to extract from the database using an extraction-filter file. The
  extraction filter that you use depends on the type of data that you want to
  extract.

### 8.2 Business and store models

One of the most significant advances in IBM WebSphere Commerce V5.5,
Business Edition is the addition of several new business models, including
Hosting, and value chains for the demand chain and supply chain models, as well
as enhancements to the B2B direct and consumer direct models.

#### 8.2.1 Business and store model overview

Table 8-1 provides a summary of the business models and corresponding
sample store model listed by the supported WebSphere Commerce Edition.

The sample stores are packaged in a store archive file for deployment. In
Table 8-1 we have listed the composite store archive name for the sample store.
In WebSphere Commerce V5.5 there is a new concept of composite or all
contents of the store archive and component store archives containing a subset
of functionality of the sample store model. We will describe this concept in
greater detail throughout this publication.

<table>
<thead>
<tr>
<th>Business model name</th>
<th>Store model name</th>
<th>Composite store archive name</th>
<th>Professional Edition</th>
<th>Business Edition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer direct (B2C)</td>
<td>FashionFlow</td>
<td>ConsumerDirect.sar</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>B2B Direct</td>
<td>ToolTech</td>
<td>B2BDirect.sar</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
IBM WebSphere Commerce V5.5 provides the architecture infrastructure to support the business models. The business models are categorized as follows:

- **Direct sales**
  - Consumer direct
  - B2B direct
- **Hosting**
- **Value chains**
  - Demand chain
  - Supply chain

**Direct sales**
Both the consumer direct and B2B direct are direct sales business models. In this section we will describe the distinguishing features of each.

**Consumer direct**
Consumer direct supports commerce transactions involving products, services, or information between businesses and consumers. Consumers typically purchase goods or services directly from a business. Consumer direct is commonly known in the industry as business-to-consumer (B2C) commerce. Figure 8-3 on page 25 depicts the interaction of consumer-oriented customers (shoppers) with a retail e-commerce site. The consumer direct model is included in the WebSphere Commerce Professional and Business Editions. All other models are only supported by the Business Edition.
In a typical consumer direct business, customers buy directly from the business, usually a retailer, as seen in Figure 8-3. The business can be a retailer, a manufacturer who sells their goods directly to consumers through their own retail outlet, or any other business that sells goods or provides services directly to consumers. For example, a business that sells to consumers directly through a catalog would be considered a consumer direct business.

Organizations that are not traditionally considered businesses, such as governments, can also be considered consumer direct businesses. Governments may provide goods and services directly to customers.

**B2B direct**

The B2B direct model supports commerce transactions involving products, services, or information between two businesses or parties. Typical B2B direct transactions occur between buyers, suppliers, manufacturers, resellers, distributors, and trading partners. Figure 8-4 depicts the interaction of business customers with a business direct e-commerce site.

In a typical B2B direct business, businesses purchase goods or services directly from another business. The selling business can be a wholesaler, a distributor, a manufacturer, or a retailer who sells to buyers from other businesses.

Organizations that are not traditionally considered businesses, such as governments and the media, can also be considered B2B direct businesses. Governments may provide goods and services directly to businesses.

**Hosting**

The hosting model supports hosting of merchants or other businesses by an Internet Service Provider (ISP) or other hosting provider.
There are two possible sides to the hosting business:

- Hosted stores
- Site that allows customers to locate the stores that are hosted by the provider (optional)

In order to manage relationships with the hosted stores, hosting models usually include a hub (known in WebSphere Commerce as a hub store). This hub provides self-provisioning tools that allow the merchant to create and administer a store, as well as tools that allow the hosting provider to manage all hosted stores.

Hosting providers typically include a store in which customers can find and access the stores hosted by the provider. Figure 8-5 depicts both merchants and customers interacting with the hosted stores of an Internet Service Provider (ISP).

As seen in Figure 8-5, the merchant enters the host’s site and creates a store that will be hosted by the site. Hosting providers often provide merchants with simple self-provisioning tools that allow the merchant to administer a hosted store. When a hosted store is open for e-business, customers can access the store via the host’s site or by entering the hosted store directly.

In this example, the customer has the option of entering the hosted store or business directly or browsing the host’s site and then being transferred to the hosted store or business.

Hosted stores are very similar to consumer direct stores. For specific differences between the two, as implemented in the WebSphere Commerce sample stores, see the *Sample Store Guide, IBM WebSphere Commerce V5.5* product guide.
Value chains
New to WebSphere Commerce Version 5.5 is the capability to enable online business transactions involving multiple enterprises. Value chains support transactions involving multiple enterprises or parties. Products, goods, services, or information are delivered through the parties of the value chain from producers to end users. A value chain also has relationship and administrative aspects; that is, you can manage the relationship of the partners or enterprises in your value chain, as well as offer some administrative services to those parties.

As a result, value chains must manage the two sides of their businesses: Their customers and direct sales, and their channel partners and suppliers. Each of these sides requires its own management channels and practices.

In order to manage their relationships with partners or suppliers, value chain business models usually include a hub (in WebSphere Commerce known as a hub store). Value chain administrators can administer the operational aspects of the value chain in the hub store, including enabling partners or suppliers to participate in the value chain, that is, registering them, setting them up, creating collaborations. Partners and suppliers can also access the hub store to complete administrative tasks such as registering users.

In order to sell directly to customers (direct sales), value chains usually include a store front, where customers can purchase their good or services directly.

WebSphere Commerce supports the transactions through, and relationship management of, the following two types of value chains:

- Demand chain
- Supply chain

Figure 8-6 on page 28 provides an overview of the partners and relationships supported by value chains.
Demand chain

A demand chain is composed of the enterprises that sell a business’s goods or services. For example, a demand chain may be composed of buyers who initiate the sales transaction, the resellers who sell the manufacturer’s goods, and the manufacturer who creates the goods. Alternatively, a demand chain may be composed of the resellers who sell a manufacturer’s goods, the manufacturer who makes the goods, and the distributors who supply the manufacturer’s goods to the resellers. Demand chains also support direct sales channels, in which the demand chain owner sells directly to customers or partners itself.

There are several types of demand chains. We have listed the following for reference:

- Demand chain hosting: Buyers, resellers, manufacturer
- Demand chain hosting: Resellers, manufacturers, distributor
- Other demand chain scenarios

Demand chain hosting: Buyers, resellers, manufacturer

The demand chain owner may host stores for its channel partners (for example, resellers or distributors). Figure 8-7 on page 29 depicts an example of a demand chain between buyers, channel partners, and manufacturer. In this example, buyers purchase goods from a manufacturer’s resellers (channel partners).
Resellers, in turn, obtain the goods from the manufacturer, via the manufacturer's hub.

**Note:** The resellers may be hosted by the manufacturer or be remote.

![Figure 8-7 Demand chain hosting (hosting for channel partners)](image)

**Demand chain hosting: Resellers, manufacturer, distributor**

In this example the manufacturer provides a hub for their channel partners including resellers. Resellers and other channel partners may be able to do several functions in this hub, including locating distributors of the manufacturer's goods.

In order to locate suppliers, the reseller may browse a product catalog in the private hub. If the desired products are available from more than one distributor, the reseller can check product availability, distributors' location, and prices for various distributors. Then, if the reseller chooses, they can split their order between several distributors. The order is then sent to the distributor, who completes the transaction and delivers the goods or services to the reseller. The reseller then sells the goods or services directly to the consumer.

The demand chain sample store is an example of this reseller, manufacturer, and distributor scenario.

**Note:** The resellers may be hosted by the manufacturer or be remote.

**Other demand chain scenarios**

There are many possible demand chain scenarios. The scenario details may change depending on the type of business being conducted. For example, if the enterprise is a manufacturer, the purpose of the hub may be to help the manufacturer's resellers locate the manufacturer's goods from several distributors. If the enterprise is a distributor, the purpose of the hub may be to help the distributor's resellers find goods or services from several different suppliers.
**Supply chain business model**

A supply chain is composed of the enterprises that provide services to a business. WebSphere Commerce provides the architectural infrastructure to support supply chains that take the form of a private marketplace.

A private marketplace provides a forum for vendors to offer their wares for sale. Buyers enter this forum and after browsing through the available options, select the appropriate goods or services.

**Note:** The private marketplace does not support competitive bidding and counter-bidding or other methods of competition.

The supply chain owner may host a store for its suppliers.

![Figure 8-8 Supply chain business model](image)

Figure 8-8 depicts a supply chain, where the buyer enters the supplier's hub to interact and browse the aggregated catalog in which products and offers from multiple suppliers are presented. The buyer can then select the desired offer or request quotes from multiple suppliers. The buy also has the option of conducting business or procuring from online suppliers directly.

### 8.2.2 Business model infrastructure and architecture

This section provides an overview of how the WebSphere Commerce infrastructure and architecture supports the business models, including the following topics:

- Organization structure
- Access control model
- Business policy framework

**Note:** The topics found in this section are a summary from the *Store Development Guide, IBM WebSphere Commerce V5.5* product guide, which we strongly recommend that you read.
**Organization structure**

The WebSphere Commerce organization structure provides a framework for the actors, or entities, in the business scenario. The framework is organized in a hierarchical structure, which mimics typical organizational hierarchies with entries for organizations and organizational units and users. The organizations and organizational units in the framework act as owners for the parts of your business. All parts of your business, including customers, administrators, stores, catalogs, and distributors, must be owned by an organization or organizational unit.

The organization structure and the access control model are closely related, in that the access control model applies access control policies to organizations rather than to individual entities (stores, customers, administrators, and so on). The policies that apply to an entity (or resource) are applied to the organizations that own the entity or resource.

---

**Figure 8-9  WebSphere Commerce organization structure**

Figure 8-9 outlines the basic WebSphere Commerce organization structure. The basic organization structure is installed during instance creation, regardless of the business model.

- **Root organization**
  
The root organization is the top level organization and is its own parent. All organizations in the WebSphere Commerce organization structure are
descendents of the root organization. The site administrators are owned by the root organization.

- **Default organization**

  The default organization is owned by the root organization. All guest customers and all customers in a consumer direct scenario belong to the default organization. Customers in a B2B direct and value chain scenario can belong to either the default organization or other organizations.

One or more other levels of organizational entities can exist beneath the parent organizational entities. You can add as many child organizational entities as necessary to support your business.

**B2B direct model and organizational structure**

We have included a diagram and description of the B2B direct business model to explain how the organization structure supports the business model. For similar information on the other business models (consumer direct, hosting, demand chain, and supply chain) refer to the *Store Development Guide, IBM WebSphere Commerce V5.5* product guide.
In order to place this business online, the entities in the preceding diagram must be assigned to the following organizations:

- **Root organization**
  All organizations in the business become descendants of the root organization. The site administrators who maintain the online site are owned by the root.

- **Default organization**
  Unlike the consumer direct organization structure, the customers are not owned by the default organization. Instead the customers are buyers who are owned by the buyer organization.

- **Buyer organization**
  Customers, known in B2B direct businesses as buyers, are assigned their own organization in the B2B direct organization structure.

- **Seller organization**
  A seller organization is created to own all the organizations that own stores. The administrators who maintain the stores’ functions (for example, customer service representatives, catalog and product managers) are termed seller administrators and are owned directly by the seller organization.

  A child organizational unit (ou), B2B direct organization, is created under the seller organization to own the store (business).

**Access control model**

WebSphere Commerce allows you to determine, through access control, which tasks a particular user, be he a customer, buyer, administrator, distributor, manufacturer, or supplier, can perform in relation to your business.

**Note:** For more detailed information on the access control model, refer to the *Store Development Guide, IBM WebSphere Commerce V5.5* and *Security Guide, IBM WebSphere Commerce V5.5* product guides.

Access control in WebSphere Commerce is composed of the following elements: Users, actions, resources, and relationships.

- Users are the people that use the system. For access control purposes, users must be grouped into relevant access groups. One common attribute that is used to determine membership of an access group is roles. Roles are assigned to users on a per-organization basis. Some examples of access groups include registered customers, guest customers, or administrative groups like customer service representatives.
Actions are the activities that users can perform on the resource. For access control purposes, actions must also be grouped into relevant action groups. For example, a common action used in a store is a view. A view is invoked to display a store page to customers. The views used in your store must be declared as actions and assigned to an action group before they can be accessed.

Resources are the entities that are protected. For example, if the action is a view, the resource to be protected is the command that invoked the view, for example, com.ibm.commerce.command.ViewCommand. For access control purposes, resources are grouped into resource groups.

Relationships are the relationships between the user and the resource. Access control policies may require that a relationship between the user and the resource be satisfied. For example, users may only be allowed to display the orders that they have created.

**Access control policies**

Access control policies authorize access groups to perform particular actions on the resources of WebSphere Commerce, as long as the users in the access group satisfy a particular relationship with respect to the resource.

WebSphere Commerce provides over three hundred default access control policies that are loaded during instance creation. These policies cover a wide range of common business activities, including order creation and processing, and trading, such as requests for quotes and contracts (Business Edition). The default policies are documented in the *Security Guide, IBM WebSphere Commerce V5.5* product guide.

**Access control groups**

In order for an access control policy to be applied to your store or site, it must belong to an access control policy group and the policy group must be subscribed by the organization that owns the resource. By default, all access control policies provided with WebSphere Commerce are assigned to policy groups. For a list of default policies provided with WebSphere Commerce, see the *Security Guide, IBM WebSphere Commerce V5.5* product guide.

Although access control policy groups are owned by organizations, they are not automatically applied to the organization. An organization must subscribe to a policy group in order for the access control policies to apply to the organization. If the organization has child organizations, all policy groups the parent subscribes to are automatically applied to the child organizations. However, if the child organization subscribes directly to a policy group, the policy groups subscribed to by the parent organization no longer apply to the child.
Basic access control structure

The WebSphere Commerce access control structure is flexible enough to support all entities in the supported business models. The diagrams in the following sections demonstrate how access control is applied to a typical example of each business model.

The basic access control structure is installed during instance creation, regardless of the business model.

The root organization owns the following default policy groups:

- Management and administration
- Common shopping
- B2C
- B2B

Note: Refer to the *Store Development Guide, IBM WebSphere Commerce V5.5* for a comparison of access control between WebSphere Commerce V5.4 and V5.5.
However, the root organization only subscribes to the management and administration policy group. As a result, these policies apply to the site administrators, who are directly under the root.

The policies in the management and administration policy group do not apply to the default organization through inheritance, as the default organization subscribes to the guest shopper management policy group. In order for the management and administration policies to apply, the default organization must subscribe to the management and administration policy group explicitly.

The default organization owns the guest shopper management policy group.

**Note:** For more detailed information on the default policy groups, see the appendix of the *Security Guide, IBM WebSphere Commerce V5.5*.

**B2B direct access control structure**

The *Store Development Guide, IBM WebSphere Commerce V5.5* includes diagrams and a description of the access control structure for each of the business models (consumer direct, B2B direct, hosting, demand chain, supply chain). This section includes a summary of the B2B direct access control structure for example purposes, as seen in Figure 8-12 on page 37.
Figure 8-12 B2B direct access control structure

Figure 8-12 depicts a basic B2B direct organization structure where the root organization owns and subscribes to the default policy groups as described in “Basic access control structure” on page 35. The B2B direct organization subscribes directly to the B2B, management and administration, and the common shopping policy groups.

The B2B direct organization also owns and subscribes to the ToolTech policy group. The ToolTech policy group contains the following policies:

- AllUsersForToolTechExecuteToolTechAllUsersViews
- RegisteredCustomersForOrgForToolTechExecuteToolTech
RegisteredCustomerViews buyers are customers that place orders in a B2Bdirect store. All buyers must be owned by a buyer organization. Typically, buyer organizations do not subscribe to any policy groups, since management and administration policies inherited from the root organization are sufficient.

Since access control policy groups are subscribed by organizational entities, if you are creating multiple stores in your site, and want to apply different access control policy groups to individual stores, you must create separate organizations to own each store.

**Business policy framework**

Business policies are sets of rules followed by a store or group of stores that define business processes, industry practices, the scope and characteristics of a store or group of stores offerings, and how the store or site interacts with customers and other business partners. For example, your site may have business policies determining when and how customers are allowed to return products to a store, or business policies that determine what payment methods your store accepts.

WebSphere Commerce provides a framework that allows you to implement your store’s business policies in your online store or site. The business policy framework consists of the following parts:

- Business policies
- Business accounts
- Contracts and service agreements
- Terms and conditions
- Business accounts

**Business policies**

In most instances, you will have predefined business policies for your business that you need to implement in your online store or site. WebSphere Commerce provides a set of business policies that you can use as is, or change to meet your needs. For more information on the default business policies provided with WebSphere Commerce, see the WebSphere Commerce Production and Development online help. For information on how to edit these business policies, see WebSphere Commerce Production and Development online help.

**Business accounts**

Business accounts define the relationship between a customer and your business. Business accounts track contracts and orders for customer organizations and configure how buyers from customer organizations shop in a store.
**Contracts and service agreements**
Before a customer or business partner (for example, resellers or distributors) can access your store, you must create a contract or service agreement that defines customer or business partner access to your store. In the WebSphere Commerce business policy framework, you create contracts for customers and service agreements for other types of business partners.

- **Contracts**: A contract with a customer defines what areas of your store the customer can access, what prices the customer will see, and for how long the customer has access to your site and those prices. All stores must contain at least one contract, as without a contract no one but internal administrators can access your store. WebSphere Commerce provides a default contract that applies to all customers shopping at a store. In WebSphere Commerce Professional Edition, the default contract is the only supported contract.

- **Service agreements**: A service agreement with a business partner (business partners may be resellers, distributors, manufacturers, suppliers, or other partners) defines your arrangement with the business partner. For example, a service agreement with a reseller may define what access the reseller has to your site, whether they can share your catalog, or whether you host a store for them. A service agreement with a distributor may define how customers to your site can receive quotes from a distributor, or how customers can access the distributors site from yours.

**Terms and conditions**
Terms and conditions define how contracts and service agreements are implemented for a particular customer or business partner. For contracts, terms and conditions may define what is being sold under the contract; the price of the items being sold; how the items are shipped to the customer; and how the customer pays for the order. For service agreements with business partners, terms and conditions may restrict the products the business partner is allowed to sell.

Terms and conditions usually reference business policies, as most aspects of a site or store operations are defined by business policies. Terms and conditions provide standard parameters for the business polices they reference. Providing parameters to the business policies allows you to modify the behavior of business policies for each contract.

**8.2.3 Store architecture**
As IT architects and IT specialists, it is great to build a scalable and secure runtime, but if you do not have an online store that customers are happy with, the rest is pointless. After all, e-commerce is focused around selling goods and services from the online store.
In this section, we explore store architecture including the assets of a store, sample store models and packaging, store data assets, customizing a store, and options for publishing a store to the runtime environment.

**Note:** For more detailed information on the WebSphere Commerce store architecture, understanding the contents of a store, and developing a store, we strongly recommend that you refer to the *Store Development Guide, IBM WebSphere Commerce V5.5* product guide.

This section includes the following topics on the WebSphere Commerce store:

- Store assets
- Store architecture
- Store packaging and models
- Store data assets and architecture
- Catalog data assets and concepts
- Tools and store data
- Customize a store
- Publish a store

**Store assets**

At the highest level, the store assets can be categorized as follows:

- Store front assets

  These are the store pages displayed to a customer. The store front content includes HTML pages, JSPs, images, and multimedia files.

  The majority of the store pages use JavaServer Page (JSP) technology. Each JSP contains HTML static content for headings, descriptions, etc. In addition, the JSPs contain JavaScript to provide client-side input checking and more sophisticated display features. JSPs also include URLs to invoke WebSphere Commerce commands and other views, as well as JSP tags and Java code for generating dynamic content. WebSphere Commerce data beans are included within the JSPs to allow access to information from the store database. The results of the access, such as product price or product attributes, are displayed on the store page (JSP).

- Back office assets (business logic)

  The back office assets contain the business logic for the store that customers do not see directly. These assets include WebSphere Commerce commands and tasks implemented through Java servlets and EJBs.
Store data
This includes all of the data related to your store, including product catalog data, tax and shipping information, payment methods, etc. In addition, this includes some configuration data used to manage your store.

Store architecture
This section describes the store architecture for the WebSphere Commerce Server instance, WebSphere Commerce Server, and store configuration scenarios.

**WebSphere Commerce Server instance**
The WebSphere Commerce Server instance is used to host a WebSphere Commerce store or multiple stores. All stores within a WebSphere Commerce instance share the same WebSphere Commerce instance database and may share some types of data such as the catalog and fulfillment methods. All stores within the instance also share the same EJB container.

When a WebSphere Commerce instance is created, the Configuration Manager creates an application server within the WebSphere Application Server for the instance. In addition, during instance creation an instance directory and subdirectories are created for instance configuration files and log files.

**WebSphere Commerce Server**
The WebSphere Commerce Server is an enterprise application deployed to its own application server that provides the infrastructure and functionality for the e-commerce solution. The store assets are deployed and hosted by the WebSphere Commerce Server (running as an application server on the WebSphere Application Server).

As noted above, a WebSphere Commerce instance can host a single store or multiple stores. We have listed several combinations of configurations for stores:

- Single store in an instance
- Multiple stores in an instance with non-shared resources
- Multiple stores in an instance with sharing resources

**Single store in an instance**
Figure 8-13 on page 42 depicts a single store published in a WebSphere Commerce instance. The three asset types are noted within Figure 8-13 on page 42, including store front, back office (business logic), and store data.
**Multiple stores in an instance with non-shared resources**

Figure 8-13 also depicts multiple stores published in a WebSphere Commerce instance. In this example, each store has its own store front, business logic, and store data.

**Multiple stores in an instance with sharing resources**

Figure 8-14 on page 43 depicts multiple stores published in a WebSphere Commerce instance with several examples of how resources can be shared. For example, stores can share store front assets, business logic, and the product catalog, or any combination of the asset types.

Multiple stores can exist in a single Stores Web module. If so, the store assets are separated using the following methods:

- **Store front assets**: Store front assets for each store in the stores Web module are stored in a separate store directory (`storedir`). For example, all store front assets for MyStore are in the MyStore directory.

- **Business logic**: The store ID is used to select the command implementation for each store, as specified in the command registry.
**Figure 8-14  Multiple stores in an instance sharing resources**

**Store packaging and models**

A store archive (or SAR) file is a zip file used for the packaging and delivery of WebSphere Commerce sample stores. A store archive can contain the following types of assets:

- Store presentation assets (JSP, images, HTML, properties files)
- Store data and configuration assets (catalog, tax, shipping, contract, flow configuration, all in XML format)
- Descriptors that describe contents and control store publishing
A store archive can be deployed or published to the runtime using the Publish Tool available from the WebSphere Commerce Administration Console. In previous releases, this functionality was provided from Store Services (discontinued).

In WebSphere Commerce V5.5 there are new types of store archives to allow for a more modular approach to store publishing. The store archive containing all assets, which is similar to previous releases, is known as a composite store archive. In addition to the composite store archives (all assets), there is now the concept of component store archives containing a subset of store function that can be published separately.

All of the store archives that are available include globalization support for the locales listed in Table 8-2. By default the store will use the locale of the WebSphere Commerce instance.

**Table 8-2  WebSphere Commerce store archive supported locales**

<table>
<thead>
<tr>
<th>Locale</th>
<th>Language and territory</th>
<th>langId</th>
</tr>
</thead>
<tbody>
<tr>
<td>en_US</td>
<td>English US</td>
<td>-1</td>
</tr>
<tr>
<td>fr_FR</td>
<td>French France</td>
<td>-2</td>
</tr>
<tr>
<td>de_DE</td>
<td>German Germany</td>
<td>-3</td>
</tr>
<tr>
<td>it_IT</td>
<td>Italian Italy</td>
<td>-4</td>
</tr>
<tr>
<td>es_ES</td>
<td>Spanish Spain</td>
<td>-5</td>
</tr>
<tr>
<td>pt_BR</td>
<td>Portugese Brazil</td>
<td>-6</td>
</tr>
<tr>
<td>zh_CN</td>
<td>Simplified Chinese China</td>
<td>-7</td>
</tr>
<tr>
<td>zh_TW</td>
<td>Traditional Chinese Taiwan</td>
<td>-8</td>
</tr>
<tr>
<td>ko_KR</td>
<td>Korean Korea</td>
<td>-9</td>
</tr>
<tr>
<td>ja_JP</td>
<td>Japanese Japan</td>
<td>-10</td>
</tr>
</tbody>
</table>

The following sections describe the store archives included with the WebSphere Commerce Business Edition:

- Consumer direct store archives
- Consumer direct basic store archive
- B2B direct store archives
- Hosting store archives
- Demand chain store archives
- Supply chain store archives
**Consumer direct store archives**

Table 8-3 provides a listing of the consumer direct store archive names and a brief description of the contents of the store archives.

<table>
<thead>
<tr>
<th>Store archive name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsumerDirect.sar</td>
<td>A sample composite store archive containing the organization structure, predefined user roles, and necessary access control policies to create consumer direct environment, plus a feature-rich working store</td>
</tr>
<tr>
<td>ConsumerDirectOrganizationStructure.sar</td>
<td>A sample store archive containing the organization structure, predefined user roles, and necessary access control policies to create a consumer direct environment</td>
</tr>
<tr>
<td>ConsumerDirectStore.sar</td>
<td>A sample store archive containing all the necessary assets to create a feature-rich working Consumer Direct store</td>
</tr>
</tbody>
</table>

**Consumer direct basic store archive**

The consumer direct basic store archive (ConsumerDirectBasicStore.sar) is a scaled down store archive to be used for the development of a store from scratch. It contains the required elements of a store archive such as one currency, one supported language, one fulfillment, store default contract, one category, one product, simple store front and shopping flow, which includes shopcart and checkout. By default, the consumer direct basic store archive is not listed from the WebSphere Commerce Administration Console under the Publish menu were other stores are listed.

**B2B direct store archives**

Table 8-4 provides a listing of the B2B direct store archive names and a brief description of the contents of the store archives.

<table>
<thead>
<tr>
<th>Store archive name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2BDirect.sar</td>
<td>A sample composite store archive containing the organization structure, predefined user roles, and necessary access control policies to create a business direct environment, plus a feature-rich working store</td>
</tr>
<tr>
<td>B2BDirectOrganizationStructure.sar</td>
<td>A sample store archive containing the organization structure, predefined user roles, and necessary access control policies to create a B2B direct environment</td>
</tr>
<tr>
<td>B2BDirectSite.sar</td>
<td>A sample store archive containing all the necessary assets to create a feature-rich working B2B direct store</td>
</tr>
</tbody>
</table>
When publishing the composite store archive (BusinessDirect.sar), all displayed items in Figure 8-15, except Root Organization and Default Organization, will be published. The Root Organization and Default Organization are created in bootstrap data. The items included in each component store archive are noted in the legend.

![Diagram](image)

**Figure 8-15  B2B direct store archives**

**Hosting store archives**

Table 8-5 provides a listing of the hosting store archive names and a brief description of the contents of the store archives.

<table>
<thead>
<tr>
<th>Store archive name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hosting.sar</td>
<td>Contains the organization structure, predefined user roles, and necessary access control policies to create a reseller hosting environment, plus the necessary assets to create a hosting solution, including a service provider site, store directory, shared catalog, and seller stores</td>
</tr>
</tbody>
</table>
Table 8-6 provides a listing of the demand chain store archive names and a brief description of the contents of the store archive.

<table>
<thead>
<tr>
<th>Store archive name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CatalogAssetStore.sar</td>
<td>A sample store archive containing all the necessary assets to create a shared catalog</td>
</tr>
<tr>
<td>HostedStoreFrontAssetStore.sar</td>
<td>Contains all the necessary assets to create a storefront</td>
</tr>
<tr>
<td>HostingHub.sar</td>
<td>Contains all the necessary assets to create the hub site</td>
</tr>
<tr>
<td>HostingOrganizationStructure.sar</td>
<td>Contains the organization structure, predefined user roles, and necessary access control policies to create a hosted environment</td>
</tr>
<tr>
<td>StoreDirectory.sar</td>
<td>Contains all the necessary assets to create a navigable directory of all stores available in the site</td>
</tr>
</tbody>
</table>

**Demand chain store archives**

Table 8-6 provides a listing of the demand chain store archive names and a brief description of the contents of the store archive.

<table>
<thead>
<tr>
<th>Store archive name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DemandChain.sar</td>
<td>A sample composite store archive containing the organization structure, predefined user roles, and necessary access control policies to create a demand chain environment, plus the necessary assets to create a demand chain solution, including a channel hub site, shared catalog, and reseller and distributor stores</td>
</tr>
<tr>
<td>CatalogAssetStore.sar</td>
<td>A sample store archive containing all the necessary assets to create a shared catalog</td>
</tr>
<tr>
<td>ChannelHub.sar</td>
<td>A sample store archive containing all the necessary assets to create a hub site</td>
</tr>
<tr>
<td>DemandChainOrganizationStructure.sar</td>
<td>A sample store archive containing the organization structure and predefined user roles to create a demand chain environment</td>
</tr>
<tr>
<td>DistributorAssetStore.sar</td>
<td>Contains all the necessary assets to support distributor proxy stores</td>
</tr>
<tr>
<td>DistributorProxyOrganizationStructure.sar</td>
<td>A sample store archive containing the organization structure and predefined user roles to create top level organization structure for the distributor proxy stores</td>
</tr>
<tr>
<td>ResellerStorefrontAssetStore.sar</td>
<td>Contains all the necessary assets to create a reseller storefront</td>
</tr>
</tbody>
</table>
Supply chain store archives

Table 8-7 provides a listing of the supply chain store archive names and a brief description of the contents of the store archives.

<table>
<thead>
<tr>
<th>Store archive name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupplyChain.sar</td>
<td>Contains the organization structure, predefined user roles, and necessary access control policies to create a supply chain environment</td>
</tr>
<tr>
<td>CatalogAssetStore.sar</td>
<td>Contains all the necessary assets to create a shared catalog</td>
</tr>
<tr>
<td>SupplierAssetStore.sar</td>
<td>Contains all the necessary assets to create a supplier storefront</td>
</tr>
<tr>
<td>SupplierHub.sar</td>
<td>Contains all the necessary assets to create a supplier hub site</td>
</tr>
<tr>
<td>SupplyChainOrganizationStructure.sar</td>
<td>Contains the organization structure, predefined user roles, and necessary access control policies to create a supply chain environment</td>
</tr>
</tbody>
</table>

Store data assets and architecture

Store data is the information loaded into the WebSphere Commerce instance database, which allows your store to function. In order to operate properly, a store must have the data in place to support all customer activities. For example, in order for a customer to make a purchase, your store must contain a catalog of goods for sale (catalog data), the data associated with processing orders (tax and shipping data), and the inventory to fulfill the request (inventory and fulfillment data).

Store data assets

Figure 8-16 on page 49 contains a summary of the store data assets. For more detailed information refer to the Store Development Guide, IBM WebSphere Commerce V5.5 product guide.
Figure 8-16  WebSphere Commerce store data assets

**Store data architecture**

Data in WebSphere Commerce stores conforms to the architecture depicted in Figure 8-17 on page 50. The store data assets can be categorized as follows:

- WebSphere Commerce Server instance data
- Core data
- Configuration data
- Managed data
- Operational data
Figure 8-17  Store data architecture

**WebSphere Commerce Server instance data**

The WebSphere Commerce Server instance data is identified as site-level information. When an instance is created, the bootstrap files (XML format) are used to populate the database. The following information is available at the site level and can be shared between multiple stores:

- Calculation usage types, device types (Web browsers, mobile device, e-mail, etc.), message types, roles, and addresses
- The default site administrator ID (user defined as instance creation)
- The default URLs, commands, and views
- The default business policies.
- The default groups and access control policies
- The language and currency supported by the instance
- The default quantity units and quantity unit conversions
- The default scheduled jobs and statecodes
- The default terms and conditions
The default organization, which can be used as the store owner
The default site organization
The default store group
The default information for staging

**Core data**
Core data represents the minimum data needed for a store. This type of data is typically packaged within the store archive file and is further categorized by organization and store.

- **Organization**: The organization core data creates the minimum data for the business model (Root Organization and Default Organization within the bootstrap data). The organization data is found in the composite store archives and the component store archive for the organization structure.
  - Organizational structure
  - Predefined user roles
  - Necessary access control policies
- **Store**: The store data is found in the composite store archives and the component store archive for the store data.
- The store identifier in the STOREENT table. This creates a store in the database.
- The default contract.
- The store identifier in the contract database tables.
- The member identifier for the organization that owns the store in the contract database tables.
- The store directory in the STORE table. The store directory is the directory in which the store’s Web assets are located.
- The nickname or identifier for the store’s address in the STADDRESS table. The nickname is unique for each store.

**Configuration data**
The following configuration data can be set at both the site and store level. The configuration data provides the Web Controller with runtime information for the appropriate commands and views to be invoked in response to each possible service request:

- URL registry entries, which map URL requests into command invocations
- Command registry entries, which basically provide the command implementation class
View registry entries, which map the view names returned by commands with view commands and related JSP templates, based both on the store ID and on the requesting device type.

**Managed data**

The following is a list of managed data created by the sellers, with read-only access for customers (note that store-specific assets have been specified):

- Business policies
- Campaigns
- Catalogs
- Contracts
- Coupons
- Currencies (store-specific)
- Customers
- Discounts (store-specific)
- E-mail activity
- Fulfillment centers
- Inventory
- Jurisdictions (store-specific)
- Languages (store-specific)
- Members
- Payment
- Prices
- Sellers
- Taxes (store-specific)
- Shipping (store-specific)
- Supported units of measure (store-specific)

**Operational data**

The following data can be created and changed, directly or indirectly, by interacting with the site. Such interactions can be performed both by customers and by sellers (note that, in this last case, operational data resulting from a particular type of interaction can be customers themselves):

- Auctions
- Contracts
- Customers
- E-mail activity
- Fulfillment
- Inventory (store-specific)
- Orders (store-specific)
- Request for quotes (RFQ)
Catalog data assets and concepts

The WebSphere Commerce database schema has a rich environment to support a very advanced product catalog. In “Store data assets and architecture” on page 48 we described the XML representation of data to be imported into the WebSphere Commerce instance database.

The primary tools used to manage catalog data in WebSphere Commerce V5.5 are the WebSphere Commerce Accelerator Product Management tool and the Loader Package.

This section defines the key catalog concepts for data stored in the WebSphere Commerce instance database. Figure 8-18 depicts the catalog hierarchy of the master catalog, catalog groups, products, and SKUs.

![Figure 8-18 Catalog hierarchy (master catalog, catalog groups, products, SKUs)](image)

**Master catalog**

The catalog object is the root of a catalog for both master catalogs and navigational catalogs. The catalog contains all the hierarchical and navigational information for the catalog. The reference identifier is used in all business objects that define the relationship of catalog groups and catalog entries. The database table for catalog is named CATALOG.
Navigational catalog
Navigational catalogs allow for organizing the products differently than defined in the master catalog for the purpose of optimizing catalog data for different types of shoppers.

Catalog group
The catalog group object is used to organize the catalog and help the shopper navigate to the products they wish to view. A catalog group may contain other catalog groups as seen in Figure 8-18 on page 53. Catalog groups are often referred to as categories. The database table for the catalog group is named CATGROUP.

Catalog entry
A catalog entry object represents merchandise in the catalog that can be ordered from the catalog. There are four different types of catalog entries, which can be identified by the value of catenttype_id in CATENTRY, including product, SKU (item), bundle, and kit.

Product
A product represents a collection of SKUs (items) that have identical attributes but can be distinguished by their attribute values. For example, a catalog may contain a shirt that is available in different sizes (small, medium, large) and colors (red, white, blue).

Attributes (defining and descriptive)
A catalog entry can have defining and descriptive attributes. Defining attributes are properties of SKUs such as color or size. Attribute values are for a color attribute and are colors such as red, white, and blue.

Descriptive attributes are simply additional descriptive information, such as cleaning information (dry clean only, etc.).

Each unique attribute value combination is used to define the item. Collectively, SKUs can be displayed as a single product with a single price, description, and image as long as a given shopper is given the opportunity to select the size and color values necessary to resolve the selection to a single SKU.

SKUs
SKUs and items are synonymous. A SKU represents a tangible unit of merchandise that can be purchased by a shopper. SKUs are products with a unique combination of defining attribute values.
**Bundle**
A bundle is a collection of catalog entries that can be added to the shopping cart with a single selection. It is typically used as a convenient method of packaging items for the shopper to order a collection of items. Bundled items can be sold separately. For example, a bundle may be a computer bundle (system, monitor, and printer). The price of the bundle is the collective price of each item.

**Pre-built kit**
A pre-built kit is a collection of catalog entries that cannot be sold separately. For example, the system component of a computer includes a motherboard, hard disk, memory, video adapter, CPU case, CD ROM, etc. Items such as a motherboard or CPU may not be offered as individual items for purchase. A pre-built kit often has its own price. A pre-built kit can be added to the shopping cart by one selection. Similar to a product, pre-built kits can have defining attributes. Once a pre-built kit is added to the shopping cart, it cannot be modified (must be removed, select attributes and add to cart). In previous releases of WebSphere Commerce, a pre-built kit was known as a package.

**Static kit**
A static kit is a group of products that are ordered as a unit. The information about the products contained in a static kit is predefined and controlled within WebSphere Commerce. The individual components within the order cannot be modified and must be fulfilled together. A static kit will back order if any of its components are unavailable.

**Dynamic kit**
A dynamic kit is an orderable SKU that consists of one or more SKUs, or components. The definition of the components that make up the kit is not known until the kit is ordered and configured, hence the name dynamic kit.

**Tools and store data**
This section provides as summary of the tools used to manage the store data, including the XML data files, before being loaded in the WebSphere Commerce instance, as well as manage the data once in the instance database.

The primary tools used to manage store data are as follows:

- WebSphere Commerce Studio
- WebSphere Commerce Loader Package
- WebSphere Commerce Administration Console
- WebSphere Commerce Organization Administration Console
- WebSphere Commerce Accelerator
**WebSphere Commerce Studio**

The IBM WebSphere Commerce Studio V5.5 includes IBM WebSphere Studio Application Developer V5. WebSphere Studio Application Developer includes an XML editor that is very useful when modify the contents of the XML data files packaged as part of the store archive.

**WebSphere Commerce Loader Package**

The WebSphere Commerce Loader Package is used to aggregate, transform, resolve, and load XML data files into the WebSphere Commerce instance database. The XML data files the Loader Package operates on are packaged with the sample store archives. In addition, the Loader Package utilities can be run via the command line or from a Java API.

When a WebSphere Commerce store is published using the WebSphere Commerce Administration Console Publish Tool, the Loader package utilities are executed in the background to resolve and load the XML data files into the WebSphere Commerce instance database.

The Loader Package includes the following utilities:

- Text Transformation Tool: Converts CSV into an XML file, and transforms XML into a WebSphere Commerce XML file.
- XSL Editor: The XSL Editor is used to create the XSL style sheets, which are used to transform (map) the XML data.
- XML Transformation Tool: This transforms the XML data using the XSL style sheets into WebSphere Commerce XML files.
- DTD Generator: This tool uses an input file containing database table names and generates either a DTD or a DTD and an XML schema to describe the select database tables.
- ID Resolver: Resolves IDs within the database with data to be loaded.
- Loader: After ID resolution, the resolved XML data files are imported into the WebSphere Commerce instance database using the Loader, also known as the MassLoad utility.
- Extractor: Used to extract data in the WebSphere Commerce instance database to an XML file.

**WebSphere Commerce Administration Console**

The Administration Console allows you to control your site or store by completing administrative operations and configuration tasks. You can also use the Administration Console to create new organizations and users, as well as assign users to roles. The Administration Console also allows you to identify which notification and messaging types will be available in your store. The
Administration Console contains the publish utility, which allows you to publish sample businesses and stores.

**WebSphere Commerce Organization Administration Console**

The Organization Administration Console allows the buyer administrators and you to control the organizations that access your site or store. Tasks that you are authorized to perform in your role are displayed on the Organization Administration Console home page menus. These tasks are based on user roles and authority levels, which are defined in XML files on your WebSphere Commerce system and are assigned by the Site Administrator by using the Administration Console.

**WebSphere Commerce Accelerator**

The WebSphere Commerce Accelerator is a workbench of online tools that allow you to create and maintain various store assets. A large portion of store data can be created and managed using the Product Management Tool found in the WebSphere Commerce Accelerator.

**Summary of tools used to manage store data**

Table 8-8 provides a summary of the WebSphere Commerce tools used to manage store data, including XML data files before load and store data found in the WebSphere Commerce instance database.

**Table 8-8  Tools used for creating store data**

<table>
<thead>
<tr>
<th>Tools to manage and customize store data</th>
<th>Core data</th>
<th>Configuration data</th>
<th>Managed data</th>
<th>Operational data</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Commerce Studio (WebSphere Studio Application Developer)</td>
<td>Applicable</td>
<td>Applicable</td>
<td>Applicable</td>
<td>In general, not applicable.</td>
</tr>
<tr>
<td>WebSphere Commerce Loader Package (data is loaded in the form of an XML file)</td>
<td>Applicable</td>
<td>Applicable</td>
<td>Applicable</td>
<td>In general, not applicable.</td>
</tr>
<tr>
<td>WebSphere Commerce Administration Console</td>
<td>Applicable * Publish store</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>WebSphere Commerce Organization Administration Console (Business Edition)</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>The organization administration console allows you to create and approve buyers.</td>
</tr>
</tbody>
</table>
Customize a store

Chances are that one of the sample stores will provide a head start to customizing a store for your needs. However, you may need to customize and add new Web assets, business logic, and your own store data. This is referred to as customizing a store. The easiest and quickest method for store customization is to start with a sample store archive that best matches your needs.

You will need to understand the programming model of WebSphere Commerce V5.5 and how to use the development tools included in WebSphere Commerce Studio V5.5 to customize your store.

<table>
<thead>
<tr>
<th>Tools to manage and customize store data</th>
<th>Core data</th>
<th>Configuration data</th>
<th>Managed data</th>
<th>Operational data</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSphere Commerce Accelerator</td>
<td>Not applicable</td>
<td>Not applicable</td>
<td>The accelerator allows you to create and edit the following data:</td>
<td>The accelerator allows you to create operational data for a customer, such as orders, and to manage the inventory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▶ Campaigns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▶ Contracts</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▶ Fulfillment</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▶ Discounts</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▶ Catalog</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>▶ Prices</td>
<td></td>
</tr>
</tbody>
</table>

Note: We strongly recommend that you read the Store Development Guide, IBM WebSphere Commerce V5.5 product guide and the WebSphere Commerce V5.5 Handbook, Customization and Deployment Guide, SG24-6969, for detailed information.

Publish a store

In WebSphere Commerce V5.5 stores are published from the Store Archive → Publish menu option found in the WebSphere Commerce Administration Console. In previous versions of WebSphere Commerce, store publishing was performed from Store Services, which has been discontinued. In addition, in previous versions the store archive was created from a sample store and then published. The generated store archive needed to be maintained. In WebSphere Commerce V5.5, the store archive is used to publish the store (there is no longer the intermediate step of creating a store archive from the sample store archive and then publishing).
8.3 Programming model

This section defines each of the following WebSphere Commerce application architecture layers, as seen in Figure 8-19:

- Database
- Business objects
- Business components
- Controls and view
- Business processes
- Store models

**Note:** For more detailed information on the WebSphere Commerce programming model we strongly recommend that you read the *Programming Guide and Tutorials, IBM WebSphere Commerce V5.5* product guide.

---

**Database**

WebSphere Commerce uses a database schema designed specifically for e-commerce applications and data requirements. The database is known as the WebSphere Commerce instance database. The instance database is created by default via the WebSphere Commerce Configuration Manager when creating the WebSphere Commerce instance.
The WebSphere Commerce instance database contains over 600 database tables, which are used to manage the operations of the WebSphere Commerce site and store. For example, information on customer profiles, taxes, orders, shipping and product catalog data are stored within the instance database. Some examples of tables found in the WebSphere Commerce instance database are USERS, ORDERS, and INVENTORY.

**Business objects**

Business objects represent entities within the commerce domain and encapsulate the data-centric logic required to extract or interpret information contained within the database. These entities comply with the Enterprise JavaBeans specification.

These entity beans act as an interface between the business components and the instance database. In addition, the entity beans are easier to comprehend than complex relationships between columns in database tables.

**Business components**

Business components are units of business logic. They perform coarse-grained procedural business logic. The logic is implemented using the WebSphere Commerce model of controller commands and task commands. An example of this type of component is the OrderProcess controller command. This particular command encapsulates all of the business logic required to process a typical order. The e-commerce application calls the OrderProcess command, which in turn calls several task commands to perform individual units of work. For example, individual task commands ensure that enough inventory is available to meet the requirements of the order, process the payment, update the status of the order, and, when the process has completed, decrement the inventory by the appropriate amount.

**Controls and view**

A Web controller determines the appropriate controller command implementation and view to be used. Implementations can be store specific. Views display the results of commands and user actions. They are implemented using JSP templates. Examples of views include ProductDisplayView (returns a product page showing relevant information for the shopper's selected product) and OrderCancelView.
Business processes
Sets of business components and views together create workflow and site flow processes that are known as business processes. Examples of business processes include:

- Creating an e-mail campaign
  This business process includes the business components and views related to all steps involved in the process of creating e-mail campaigns.

- Preparing an online catalog
  This business process includes the business components and sub-processes related to creating an online catalog. This includes designing the catalog, loading catalog data, creating merchandising associations, and setting pricing information.

Store models
When gathered together, the lower layers of the diagram make up e-commerce business models. Each of the business models supported in WebSphere Commerce include a sample store model. The WebSphere Commerce Business Edition includes the following store models: Consumer direct, B2B direct, hosting, demand chain, and supply chain.

8.3.1 WebSphere Commerce Server framework
This section defines the component architecture of the WebSphere Commerce common server runtime, also known as the WebSphere Commerce Server framework.

The WebSphere Commerce common server runtime provides a framework in which the commerce applications are deployed and executed. The framework is based on the Java 2 Enterprise Edition (J2EE) platform architecture. Such a framework provides the capability to handle system and user requests and to perform the corresponding actions in order to fulfill them.
Figure 8-20  WebSphere Commerce common server runtime component architecture
The major components of the common server runtime or framework are listed as follows and depicted in Figure 8-20 on page 62:

- Servlet engine
- Protocol Listeners
- Adapter manager
- Adapters
- Web controller
- Commands
- Entity beans
- Data beans
- Data Bean Manager
- JavaServer Page (JSP) templates
- WebSphere Commerce <instance_name>.xml configuration file

**Servlet engine**
The servlet engine is the part of the WebSphere Application Server runtime environment that acts as a request dispatcher for HTTP requests. The servlet engine manages a pool of threads to handle requests. Each inbound request is executed on a separate thread. The request is then sent to the appropriate protocol listener, in this case the HTTP request servlet.

**Protocol Listeners**
Protocol Listeners receive inbound requests based on the protocols and dispatch them to the adapter manager to find the appropriate adapter for the type of requesting device. WebSphere Commerce includes the HTTP request servlet and MQ listeners.

**HTTP request servlet**
The HTTP request servlet handles incoming HTTP requests. When the HTTP request servlet is initialized, it reads the adapter section within the <instance_name>.xml configuration file to load the adapter settings.

When the HTTP request servlet receives a URL request from the servlet engine, it passes the request to the adapter manager. The adapter manager analyzes the contents of the HTTP header for the User Agent to determine the appropriate adapter to send the request to.

**MQ listener**
The MQ listener receives requests for inbound XML-based MQ messages from remote programs and then sends the request to the MQ adapter to be processed.
**Request device types and sources**

WebSphere Commerce commands can be invoked as the result of requests from various device types such as the following:

- Web browser client
- Mobile device with a Web browser (mobile phone, PDA, etc.)
- B2B application sending XML messages using MQ
- Procurement system sending requests using XML over HTTP
- WebSphere Commerce scheduler executing a background job

**Adapter manager**

The adapter manager analyzes the contents of the HTTP header for the User Agent to determine the appropriate adapter to send the request to.

**Adapters**

Adapters are device-specific components that enable WebSphere Commerce to perform processing functions before passing on the request to the Web controller.

Adapters are responsible for the following tasks:

- Instructing the Web Controller to process the request in a manner specific to the type of device
- Translating the message from the device-specific format into a WebSphere Commerce common format (that is, a CommandProperty object)
- Providing device-specific session persistence

Adapters provided by WebSphere Commerce are as follows:

- HTTP Browser adapter
- HTTP PVC adapter
- Program adapter
- Scheduler adapter

**HTTP Browser adapter**

The HTTP Browser adapter provides support for requests to invoke WebSphere Commerce commands received from Web browser clients (HTTP).

**HTTP PVC adapter**

The HTTP PVC adapter is an abstract adapter class that can be used to develop specific PVC adapters such as a WAP PVC adapter or Portal adapter.
**Program adapter**
The program adapter provides support for remote programs to invoke WebSphere Commerce commands. The program adapter receives requests and uses a message mapper to convert the requests into CommandProperty objects. After conversion, the program adapter uses the CommandProperty object and executes the request.

**Scheduler adapter**
The scheduler adapter provides support for WebSphere Commerce commands that are run as background jobs.

**Note:** If required, the adapter framework can be extended to create new PVC adapters or to create new adapters that connect to new protocol listeners.

**Web controller**
The WebSphere Commerce Web controller is an application container that follows a design pattern similar to that of an EJB container. This container simplifies the role of commands by providing such services as session management (based upon the session persistence established by the adapter), transaction control, access control, and authentication.

The Web controller also plays a role in enforcing the programming model for the commerce application. For example, the programming model defines the types of commands that an application should write. Each type of command serves a specific purpose. Business logic must be implemented in controller commands and view logic must be implemented in view commands. The Web controller expects the controller command to return a view name. If a view name is not returned, an exception is thrown.

For HTTP requests, the Web controller performs the following tasks:

- Begins the transaction using the UserTransaction interface from the javax.transaction package.
- Gets session data from the adapter.
- Determines whether the user must be logged on before invoking the command. If required, it redirects the user’s browser to a logon URL.
- Checks if secure HTTPS is required for the URL. If it is required but the current request is not using HTTPS, it redirects the Web browser to an HTTPS URL.
- Invokes the controller command and passes it the command context and input properties objects.
If a transaction rollback exception occurs and the controller command can be retried, it retries the controller command.

A controller command normally returns a view name when there is a view command to be sent back to the client. The Web controller invokes the view command for the corresponding view. There are a number of ways to form a response view. These include redirecting to a different URL, forwarding to a JSP template, or writing an HTML document to the response object.

- Saves the session data.
- Commits the session data.
- Commits the current transaction if it is successful.
- Rolls back the current transaction in case of failure (depending upon circumstances).

**Commands**

WebSphere Commerce commands are beans that contain the programming logic associated with handling a particular request. There are four main types of WebSphere Commerce commands:

- Controller command
- Task command
- Data bean command
- View command

**Controller command**

A controller command encapsulates the logic related to a particular business process. Examples of controller commands include the OrderProcessCmd command for order processing and the LogonCmd that allows users to log on. In general, a controller command contains the control statements (for example, if, then, else), and invokes task commands to perform individual tasks in the business process. Upon completion, a controller command returns a view name. The Web controller then determines the appropriate implementation class for the view command and executes the view command.

**Task command**

A task command implements a specific unit of application logic. In general, a controller command and a set of task commands together implement the application logic for a URL request. A task command is executed in the same container as the controller command.
Data bean command
A data bean command is invoked by the data bean manager when a data bean is instantiated. The primary function of a data bean command is to populate the data bean with data.

View command
A view command composes a view as a response to a client request. There are three types of view commands:

- Redirect view command: This view command sends the view using a redirect protocol, such as the URL redirect. A controller command should return a view command in this view type to return a view using a redirect protocol. When a redirect protocol is used, it changes the URL stacks in the browser. When a reload key is entered, the redirected URL executes instead of the original URL.
- Direct view command: This view command sends the response view directly to the client.
- Forward view command: This view command forwards the view request to another Web component, such as a JSP template.

There are three ways in which a view command can be invoked:

- A controller command specifies a view command name when the request has successfully completed.
- A client requests a view directly.
- A command detects an error and an error task must be executed to process the error. The command throws an exception with a view command name. When the exception propagates to the Web controller, it executes the error view command and returns the response to the client.

Entity beans
Entity beans are the persistent, transactional commerce objects provided by WebSphere Commerce. If you are familiar with the commerce domain, entity beans represent WebSphere Commerce data in an intuitive way. That is, rather than having to understand the whole database schema, you can access data from an entity bean that more closely models concepts and objects in the commerce domain. You can extend existing entity beans. In addition, for your own application-specific business requirements, you can deploy entirely new entity beans. Entity beans are implemented according to the Enterprise JavaBeans (EJB) component model.
Data beans
Data beans are Java beans that are primarily used by Web designers. Most commonly, they provide access to a WebSphere Commerce entity. A Web designer can place these beans on a JSP template, allowing dynamic information to be populated on the page at display time. The Web designer need only understand what data the bean can provide and what data the bean requires as input. Consistent with the theme of separating display from business logic, there is no need for the Web designer to understand how the bean works.

Data Bean Manager
WebSphere Commerce data beans inserted into JSP templates allow for the inclusion of dynamic content in the page. The data bean manager activates the data bean so that its values are populated when the following line of code is inserted into the page:

```java
com.ibm.commerce.beans.DataBeanManager.activate(data_bean, request)
```

Where `data_bean` is the data bean to be activated and `request` is an HttpServletRequest object.

JavaServer Page (JSP) templates
JSP templates are specialized servlets that are typically used for display purposes. Upon completion of a URL request, the Web controller invokes a view command that invokes a JSP template. A client can also invoke a JSP template directly from the browser without an associated command. In this case, the URL for the JSP template must include the request servlet in its path, so that all of the data beans required by a JSP template can be activated within a single transaction. The request servlet can forward a URL request to a JSP template and execute the JSP template within a single transaction. The data bean manager rejects any URL for a JSP template that does not include the request servlet in its path.

WebSphere Commerce <instance>.xml configuration file
The WebSphere Commerce <instance>.xml configuration file contains the configuration settings for the WebSphere Commerce instance, and is read when the HTTP request servlet is initialized.

8.3.2 Application flow of an HTTP request
This section contains a description of the application flow of an HTTP request from a Web browser client to the WebSphere Commerce node to explain how the components interact.
Figure 8-21   Application flow of an HTTP request
The following information corresponds to the detailed flow of an HTTP request from a Web browser client product category display, as seen in Figure 8-21 on page 69:

1. The request is directed to the servlet engine by the WebSphere Application Server plug-in.
2. The request is executed in its own thread. The servlet engine dispatches the request to the HTTP request servlet protocol listener.
3. The HTTP request servlet protocol listener passes the request to the adapter manager.
4. The adapter manager determines which adapter is capable of handling the request and then forwards the request to the appropriate adapter. For example, if the request came from a Web browser, the adapter manager forwards the request to the HTTP Browser adapter.
5. The adapter passes the request to the Web Controller.
6. The Web Controller determines which command to invoke, by querying the command registry.
7. Assuming that the request requires the use of a controller command, the Web Controller invokes the appropriate controller command.
8. Once a controller command begins execution, there are a few possible paths:
   - The controller command can access the database using an access bean and its corresponding entity bean.
   - The controller command can invoke one or more task commands. Then task commands can access the database, using access beans and their corresponding entity beans (shown in 8c).
9. Upon completion, the controller command returns a view name to the Web Controller.
10. The Web Controller looks up the view name in the VIEWREG table. It invokes the view command implementation that is registered for the device type of the requester.
11. The view command forwards the request to a JSP template.
12. Within the JSP template, a data bean is required to retrieve dynamic information from the database. The data bean manager activates the data bean.
13. The data bean manager invokes a data bean command, if required.
14. The access bean from which the data bean is extended accesses the database using its corresponding entity bean.
15. The JSP writes a response, which is returned to the Web server. The Web server in turn relays that response to the Web browser client for the display of the category display page.

### 8.3.3 Customizing application assets

Each of the IBM WebSphere Commerce V5.5 editions and corresponding IBM WebSphere Commerce Studio V5.5 editions include sample stores to speed application development. This section describes the types of application assets that you may need to customize using WebSphere Commerce Studio and provides a matching of skill level needed for the customization.

#### Asset types to customize and development tooling

In combination with the rich WebSphere Commerce framework described in “WebSphere Commerce Server framework” on page 61 and the extremely powerful WebSphere Commerce Studio development tooling, virtually any application customization is possible.

We have categorized the application assets into the following categories:

- Store front assets
- Back office assets (business logic)
- Data files

#### Store front assets

Store front asset customization (new or modified) is primarily done for look and feel of the store pages. For example, if you started with the B2B direct sample store (ToolTech), you would most definitely change the store logo and headings on the store pages. This type of development is commonly performed by a Web developer (Web designer).

Store front asset types include the following:

- Static HTML pages
- JavaServer Pages (JSPs)
- Images
- JavaScript

Within WebSphere Commerce, these assets types are contained within the sample store archives.

#### Back office assets (business logic)

Back office assets (new or modified) contain the business logic. Examples of back-office assets types include:

- Commands
Beans enable the store front JSPs (containing data beans) to interact with the instance database to allow for the storage, retrieval, and manipulation of data.

**Data files**
The sample store archives contain many XML data files. Some of the data files are used for site and store configuration for such things as taxes and shipping. There are also many XML data files used to populate the WebSphere Commerce catalog. The sample store includes sample product catalog data that must be updated or removed for your customized store.

**Matching skills to customization needs**
A commerce store can have varying levels of customization. The skills required depend on this level of customization. The level of customization affects both the development time and skills needed to develop a store. We have categorized the customization requirements into three levels: Basic, intermediate, and advanced. For each level we describe the tasks involved and list the IT specialists' roles to complete the tasks.

**Basic customization**
Basic store customization normally involves the following:

- **Catalog creation**
  This involves creating a catalog and dividing the catalog into logical catalog groups or categories. The store's products are then assigned to catalog groups (categories).

- **Web page design**
  Web page design involves the creation of graphics, static HTML pages, and JSPs for a customized look and feel for the store using the existing WebSphere Commerce commands.

  The roles include site administrator and Web developer.

**Intermediate customization**
Intermediate store customization includes the elements of a basic customization plus the following tasks:

- **Custom command development**
  Custom commands, which are Java classes, are used for flow control and to implement business logic. They may extend existing commands or be created as new commands. WebSphere Commerce commands are developed in WebSphere Commerce Studio (WebSphere Studio Application Developer).
Custom data bean development

The development of data beans is required to represent dynamic content to be displayed in JSPs. The dynamic content is populated at runtime most often from the database. Data beans are developed in WebSphere Commerce Studio (WebSphere Studio Application Developer).

The roles include site administrator, Web developer, and Java programmer.

Advanced customization

Advanced store customization includes the elements of an intermediate customization plus the following tasks:

- Database customization
  The alteration of the existing database schema may be needed to meet the specialized needs of a business.

- EJB development
  The development of persistent Java objects to represent data in the database. For example, if you need to add a table to the database you will need to create an EJB to query and update that table. EJBs are developed in WebSphere Commerce Studio (WebSphere Studio Application Developer).

- Data access bean development
  The data access bean is developed in Java and uses EJBs to access data from the database. Data access beans are developed in WebSphere Commerce Studio (WebSphere Studio Application Developer).

The roles include site administrator, Web developer, Java programmer, database schema owner, and database administrator.
IBM WebSphere Portal V5.0

Portals serve as a simple, unified access point to Web applications. Portals also do much more: They provide valuable functions like security, search, collaboration, and workflow. A portal delivers integrated content and applications, plus a unified, collaborative workplace. Indeed, portals are the next-generation desktop, delivering e-business applications over the Web to all kinds of client devices.

A complete portal solution should provide users with convenient access to everything they need to get their tasks done anytime, anywhere, in a secure manner. IBM's vision is that portals are the key to providing a personalized, relevant Web experience, enabling users to readily find what they need in a highly interactive and personal way. That is, portals provide the tools and user interface to access information and applications, and to manage the selection and personalization of content.

WebSphere Portal leads the Business Portals part of the WebSphere Platform. It provides an extensible framework for interacting with enterprise applications, content, people, and processes. Self-service features allow end users to personalize and organize their own view of the portal, to manage their own profiles and to publish and share documents with their colleagues.

WebSphere Portal provides additional services such as single sign-on, security, document management, Web content publishing, search, personalization, collaboration services, enterprise application integration, support for mobile devices, and site analytics.
9.1 WebSphere Portal architecture

WebSphere Portal is the industry’s most comprehensive portal solution, and represents the de-facto standard e-business architecture. WebSphere Portal integrates both IBM and business partner technologies to realize this architecture. IBM is also extending its portal offering to create additional products that deliver highly personalized and context-sensitive applications, accessible from any device, anytime.

![WebSphere Portal Architecture Diagram]

Figure 9-1  Websphere Portal Architecture

As e-business applications enter the on-demand era, WebSphere Portal leads the way with its concepts of delegated administration, cascading page layouts, portal federation through Web services, advanced portlet application concepts, business process integration, knowledge management, document management with integrated productivity components, and advanced personalization. In complementary offerings, additional pervasive computing functions are enabled, such as intelligent notification, offline browsing, and data synchronization.

9.2 WebSphere Portal product editions

WebSphere Portal is available in several editions, each designed to provide the infrastructure you need to build and deploy highly scalable portals. All offerings share a common framework (the portal server) plus additional products and services. The portal server provides common services such as application
connectivity, integration, administration, and presentation that are required across portal environments.


**Table 9-1  Portal Enable**

<table>
<thead>
<tr>
<th>WebSphere Portal Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Portal Server</strong></td>
</tr>
<tr>
<td><strong>Personalization</strong></td>
</tr>
<tr>
<td><strong>Content Publisher</strong></td>
</tr>
<tr>
<td><strong>Portal Document Manager</strong></td>
</tr>
<tr>
<td><strong>Productivity Components</strong></td>
</tr>
<tr>
<td><strong>WebSphere Translation Server</strong></td>
</tr>
<tr>
<td><strong>WebSphere Studio Site Developer</strong></td>
</tr>
<tr>
<td><strong>Collaborative Components</strong></td>
</tr>
</tbody>
</table>

**Table 9-2  Portal Extend**

<table>
<thead>
<tr>
<th>WebSphere Portal Extend</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All products in Enable</strong></td>
</tr>
</tbody>
</table>
### Table 9-3 Portal Express

<table>
<thead>
<tr>
<th><strong>WebSphere Portal Express</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collaboration Center</strong></td>
<td>A set of three fully integrated pre-configured, ready-to-use collaborative &quot;portlets&quot; that can be used right out of the box, providing instant value from e-workplace computing. These include online directory services, instant messaging, and access to team rooms.</td>
</tr>
<tr>
<td><strong>Lotus Extended Search</strong></td>
<td>Provides parallel, distributed, heterogeneous searching capability across Lotus Notes databases, legacy data stores, Web search sites, and other sources.</td>
</tr>
<tr>
<td><strong>Tivoli Web Site Analyzer</strong></td>
<td>Analyzes Web site usage logs to reveal information that can be used to improve the portal for better user experience.</td>
</tr>
<tr>
<td><strong>Sametime</strong></td>
<td>Instant messaging, presence awareness, and Web conferencing services.</td>
</tr>
<tr>
<td><strong>QuickPlace</strong></td>
<td>Web-based solution for creating team workspaces for collaboration. Features include discussions, document collaborations, coordinate plans, tasks, and resources.</td>
</tr>
<tr>
<td><strong>Portal Server</strong></td>
<td>Provides presentation, user management, security and other services for constructing the portal site.</td>
</tr>
<tr>
<td><strong>WebSphere Studio Site Developer</strong></td>
<td>Professional developer tools for creating, testing, debugging, and deploying portlets, servlets, and other assets related to portals and Web applications.</td>
</tr>
<tr>
<td><strong>Portal Document Manager</strong></td>
<td>Centralized document storage, organization, and version management services.</td>
</tr>
<tr>
<td><strong>Collaborative Components</strong></td>
<td>Java APIs that provide the building blocks for integrating the functionality of Domino, Lotus Instant Messaging (Sametime), Lotus Team Workplaces (QuickPlace), and Lotus Discovery Server into portals and portlets.</td>
</tr>
<tr>
<td><strong>Productivity Components</strong></td>
<td>In-line view and edit capabilities for rich text, spreadsheet and presentation content.</td>
</tr>
</tbody>
</table>
Customers choosing one of the WebSphere Portal offerings can realize tangible business and technical benefits:

- Revenue benefits, as a result of tighter relationships with customers or partners, work force productivity, innovation, and reduced cycle times
- Operational cost reduction, as a result of operational efficiency, better information flow and knowledge, and consistent infrastructure
- Increased employee productivity and improved decision making because of access to more relevant information, and a single access point to applications and collaboration tools
- Better security and single sign-on, resulting in fewer passwords to administer and better user experience
- Reduced training costs, resulting from common presentation and a consistent user interface
- Unification of applications, giving them a longer useful life, with new ways of accessing them through desktop and pervasive devices

The WebSphere Portal offerings allow an e-business to quickly capitalize on its digital and human assets while presenting a first-class Web experience to its employees, partners, and customers.

This document is intended to help customers, independent software vendors, and application architects plan their use of WebSphere Portal. It explains portal applications, content, security, user management, administration, document management, search, personalization, collaboration, and more.
9.3 Portlets

Portlets are the heart of a portal. The term “portlet” refers to a small portal application, usually depicted as a small box in the Web page. Portlets are reusable components that provide access to applications, Web-based content, and other resources. Web pages, Web services, applications, and syndicated content feeds can be accessed through portlets. Companies can create their own portlets or select from a catalog of portlets created by IBM and by IBM business partners.

Any particular portlet is developed, deployed, managed, and displayed independent of other portlets. Administrators and end users create personalized portal pages by choosing and arranging portlets, resulting in Web pages that present content tailored for individuals, teams, divisions, and organizations.

The portal server already includes a rich set of standard portlets for storing and sharing documents, displaying syndicated content, performing XML transformation, accessing existing Web pages, Lotus Notes and Microsoft Exchange productivity applications, Sametime instant messaging, and Lotus QuickPlace team rooms. The following figure shows just a few of the portlets that are available from IBM.
An extensive third-party partner program makes many more portlets available from the portal catalog download site at:

http://www.ibm.com/software/genservers/portal/portlet/catalog
9.3.1 Portlet applications

Portlets are more than simple views of existing Web content. A portlet is a complete application, following a standard model-view-controller design. Portlets have multiple states and view modes, plus event and messaging capabilities.

Portlets run inside the portlet container of a portal server, similar to a servlet running on an application server. The portlet container provides a runtime environment in which portlets are instantiated, used, and finally destroyed. Portlets rely on the portal infrastructure to access user profile information, participate in window and action events, communicate with other portlets, access remote content, look up credentials, and store persistent data.
Generally, portlets are administered more dynamically than servlets. For example, portlet applications consisting of several portlets can be installed or removed while the server is running. Administrators can change the settings and access rights of a portlet while the portal is running, even in a production environment.

### 9.3.2 Portlet modes

Portlet modes allow a portlet to display a different user interface, depending on the task required of the portlet. A portlet has several modes of display, which can be invoked by icons on the portlet title bar: View, help, edit and configure.

A portlet is initially displayed in its view mode. As the user interacts with the portlet, it may display a sequence of view states, such as forms and responses, error messages, and other application-specific states.

Help mode is used to provide user assistance for the portlet. Edit mode provides a page for users to change the portlet settings. For example, a weather portlet might provide an edit page for users to specify their location. Users must be logged into the portal to access edit mode. If configure mode is supported by a portlet, it provides a page for portal administrators to configure portlet settings that are shared by instances of that portlet.

Each portlet mode can be displayed in normal, maximized, or minimized states. When a portlet is maximized, it is displayed in the entire body of the portal page, replacing the view of other portlets. When a portlet is minimized, only the portlet title bar is displayed on the portal page.

### 9.3.3 Portlet API

Portlets are a special subclass of HttpServlet, with properties that allow them to easily plug into and run in the portal server. Portlets are assembled into a larger portal page, with multiple occurrences of the same portlet displaying different data for each user. Portlets rely on the portal infrastructure to access user profile information, participate in window and action events, communicate with other portlets, access remote content, look up credentials, and store persistent data. The portlet API provides standard interfaces for these functions.

The portlet API defines a common base class and interfaces for portlets, in order to cleanly separate the portlet from the portal infrastructure. In most respects, the portlet API is an extension of the servlet API, except that it restricts certain functions to a subset that makes sense for portlets running in the context of a portal. For example, unlike servlets, portlets may not send errors or redirects as a response. This may only be done by the portal itself, which controls the overall response page.
The markup fragments that portlets produce may contain links, actions, and other content. The Portlet API defines URL rewriting methods that allow portlets to transparently create links, without needing to know how URLs are structured in the particular portal.

### 9.3.4 Portlet performance

Since portlets are servlets, similar reentrancy and performance considerations apply to both. A single portlet instance (that is, a single instance of the portlet's Java class) is shared among all requesters. There are a limited number of threads that process portlets and servlets, so it is important for each portlet to do its job as quickly as possible so that response time for the whole page is optimized.

Just as with servlet programming, you should consider optimizations such as limiting the use of the synchronized methods, limiting the use of expensive string operations, avoiding long-running loops, and minimizing the number of objects that are created. Another optimization is to use JavaServer Pages for rendering the portlet's views; in general, views created with JavaServer Pages are faster than views created with XSL.

Usually, many portlets are invoked in the course of handling a single request, each one appending its content into the overall page. Some portlets can be rendered in parallel, so that the portal server assembles all the markup fragments when all the portlets finish or time out. This improves the performance of portlets that access remote data by HTTP or SOAP requests. However, not all portlets are thread-safe; for example, portlets that access protected resources may not be run in parallel. The portlet deployment descriptor indicates whether the portlet is considered thread-safe; portlets that are not thread-safe will be rendered sequentially.

Portlet output can also be cached. The caching policies are configured in the portlet deployment descriptor, including an expiration time and whether or not the portlet markup can be shared among users or is user specific.

### 9.3.5 Standards

As portals continue to evolve into the new desktop and integration standard, IBM is leading efforts to standardize the application programming interfaces between portals and other applications. In particular, the Java Community Process (JCP) and the Organization for the Advancement of Structured Information Standards (OASIS) are working cooperatively to standardize the Java and XML technology needed to link portals to disparate applications.
JSR 168 is co-led by IBM and Sun, and seeks to achieve interoperability between local portlets and portal servers. IBM initiated the Portlet API, and will be responsible for providing an open source reference implementation at the Apache Jakarta project. Sun will edit the specification and will provide the freely accessible Compliance Test Kit. IBM will ship a JSR 168 compliant portlet container to support operation of JSR 168 portlet applications with Websphere Portal Version 5 soon after that JSR specification is finalized.

OASIS’ Web Services for Remote Portals (WSRP) Technical Committee, chaired by IBM, has created an XML and Web services standard that will allow the interoperability of visual, user-facing services with portals or other Web applications. IBM will incorporate support for WSRP 1.0 applications with Websphere Portal Version 5 soon after the WSRP 1.0 specification is finalized.

9.3.6 Struts

Struts is a Jakarta open source project that provides a framework based on the Model-View-Controller (MVC) pattern. It allows developers to efficiently separate the application’s business logic from its presentation. Struts enforces sequence of pages and actions and provides form validation functions.

This release of WebSphere Portal provides support for using the Struts 1.1 framework to build portlets. In order to work in portlets, specific technical details must be observed when using Struts. For example, when used in portlets, a Struts action should not write to the response object, and should not create header elements like HEAD and BODY tags. The Struts application must be packaged with several replacement jar files that make URLs, forward actions, and include actions work properly in a portal environment.

9.3.7 Portlet cooperation

The portal server provides a mechanism for portlets to communicate with each other, exchanging data or other messages. In a production portal, portlet communication could be used to copy common data between portlets. This saves redundant typing by the user and makes the portal easier to use. For example, one portlet might display information about accounts while a second portlet displays information about transactions that have occurred for one of the accounts over the last 30 days. To do this, the transactions portlet needs to obtain the corresponding account information when it displays the transaction details.

This is accomplished by communication between the two portlets, using portlet actions and portlet messages. In this example, the account portlet creates a portlet action and encodes it into the URL that is rendered for displaying
transactions. When the link is clicked, the action listener is called, which then sends a portlet message to send the necessary data.

Programmatic messaging helps unify portlet applications that access different backend applications. However, this is relatively static, and requires planning and design work in advance. The portlets exchanging messages must already know about each other in order to make the interchange work. Next discuss more flexible means of portlet cooperation.

### 9.3.8 Brokered cooperation

Brokered cooperation allows independently developed portlets to exchange information. Portlets register their intent to co-operate with a broker, which facilitates the exchanges at runtime. The broker works by matching data types between the sources in one portlet with the actions of another portlet. When the types match, a transfer is possible, and the broker allows the user to trigger the transfer through a pop-up menu. The term *click-to-action* is used for such menu-driven, brokered data exchange.

![Click-to-action portlets](image)

*Figure 9-5  Click-to-action portlets*

The objective of the click-to-action portlets is to increase the productivity of portal users working with multiple portlets by easily enabling them to send information
from one portlet to another. For example, users can click information that is displayed in one portlet and transfer that information to another portlet. The portlet receiving the information processes it and updates its display.

Click-to-action automatically matches the portlet information sources and possible actions based on their data type compatibility. Click-to-action does not rely on drag-and-drop or other non-standard browser features. A unique advantage of click-to-action is that it is designed to work in different browsers, making it more accessible to users.

An extension of this idea, cooperative portlets, is available with Websphere Portal Version 5. Using cooperative portlet capabilities, portlets can be pre-wired by administrators, so that they exchange data automatically. Data transfer along the wires is achieved using the same broker as click-to-action. In addition to saving the extra step of having the user click the data source to select a target, because portlets are explicitly wired together, greater flexibility is possible in terms of matching brokered values.

Figure 9-6  Pre-wired cooperating portlets
9.3.9 Discoverable services

The portlet API provides an interface to enable dynamic discovery of available services. Each service is registered in a portal configuration file, and is accessed from the PortletContext.getService() method, which looks up the factory for the service, creates the service, and returns it to the portlet. This makes services available to all portlets without having to package the service code with the portlet. The implementation of such a service can be exchanged or enhanced transparently, without affecting the portlet.

The portal server provides discoverable services for its credential vault, for managing persistent TCP/IP connections, and for managing the portal's content repository. New services may be implemented by portal developers, such as location, notification, content access, or mail services.

9.3.10 Tools

Portlets can be grouped together in a portlet application. Portlet applications are distributed and deployed using Web archive (WAR) files. There are portlet-specific extensions to the standard Web application deployment descriptor.

WebSphere Studio Site Developer provides an excellent development, test, and debug environment for portlet applications. You can implement the Java classes for portlet classes, and also test and debug the Java code. Studio also provides tools for creating JavaServer Pages, HTML pages, images, and other related portal resources. The portal toolkit provides plugins to help build the portlet deployment descriptor and package it into a WAR file, and then easily deploy it to the runtime portal server.
9.4 Content and search

When companies deploy portals, they want to view, organize, share, and find information from various sources and in various formats. WebSphere Portal has the most complete content story of any portal, providing support for syndicated content, document management, Web content management, and integration with the leading Web content management systems, advanced personalization, and search portlets that include categorization and summarization capabilities.

9.4.1 Syndicated content

A key concept related to portal technology is syndication, which is about delivering fresh, personalized, and filtered content and services from multiple sources to subscribers. Typically, the content is related to news, finance, and entertainment. Portal partners include popular content providers, such as
Pinnacor, Financial Times, Moreover, YellowBrix, Hoovers, Factiva, NewsEdge, MediaApps, DataMonitor, and others.

Figure 9-8 Syndicated content

Companies are embracing syndication concepts and standards to automate the publishing of electronic catalogs and other internal information, and to make this information available to workers through enterprise portals. A popular and useful format for syndicated news and entertainment content is Rich Site Summary (RSS). Content can be published directly from the content management system into Rich Site Summary and Open Content Syndication (OCS) channels, where it can easily be displayed by the portal server's built-in RSS portlet. This self-syndication concept defines a procedure for editing, managing, and publishing your own sources of content.

9.4.2 Web content management

Web content management deals with creating, approving, and publishing Web content from content creators to Web servers. The steps of this process include defining content types, roles, publication options, destination specifications, and workflow processes. There are many content management vendors in the marketplace today, including Lotus Workplace Content Development (formerly Aptrix), Interwoven, Vignette, Documentum, FatWire, Stellent, and others.

Although each of these products works differently, generally, they are designed to create, maintain, and publish collections of structured and unstructured content that can be made available to users via the portal.

To illustrate how this is done, WebSphere Portal includes integration kits that illustrate specific steps of how to publish RSS content from several of the Web content management products. They are available through the portlet catalog. Content contribution and approval operations of the Web content management system can also be accessed through portlets that are provided by their
respective companies. These portlets provide a user interface into various aspects of the content management process, such as content submission, workflow management, content approval, and even staging or publishing.

9.4.3 WebSphere Portal Content Publishing

The WebSphere Portal Content Publishing (WPCP) tool is intended for users that need to contribute content to a Web site on a regular or occasional basis. This includes business users creating template-driven content such as press releases or product information, and graphic artists creating and editing artwork using their favorite tools. Content Publishing supports content contribution via templates or forms, as well as the contribution of files such as images, HTML, or JSPs, which are created and edited using popular tools such as HomePage Builder, Dreamweaver, FrontPage, PhotoShop, or Word.

WPCP provides a Web browser interface that enables users to contribute content to a Web site in an easy-to-use, quick, and controlled manner. This allows large teams to work together, jointly where necessary, but with enough isolation that they are not constantly tripping over each others’ changes.

Figure 9-9   Content publishing tools
You can use the content publishing tools to manage content that is served through portlets within the portal server. WPCP can also be used to author personalization rules and campaigns in order to target content to specific groups or users and to report on the content usage within your portal.

### 9.4.4 Workflow

WPCP manages the task lists of each user, maintains access control over what content each user can see or change, and coordinates the approval and publishing process when the content is ready.

![Content publishing workflow](image)

*Table 9-5  Content publishing workflow*

WPCP supports a lightweight workflow option for prototyping and testing, and includes an embedded engine, WebSphere Process Choreographer, to manage production workflow operations. To define a workflow process, use the workflow process design tool of the corresponding workflow engine. Each step in the process is assigned to a specific user role, such as an editor or an approver.
When the process starts, a work item is put on the work queue of each person in the specified role. As users claim activities and complete the steps, the work is put on the next queue defined by the workflow process.

9.4.5 Enterprise content management

For managing and searching large collections of content created and used by a variety of applications, you need an enterprise content management solution such as IBM Content Manager. WebSphere Portal includes many functions that exploit IBM Content Manager, including WPCP.

Additional search portlets from leading search vendors (Verity, Inktomi, Autonomy) are also available and can be found on the portal catalog.

9.4.6 Document management

WebSphere Portal provides a document management capability. Portal Document Manager (PDM) allows authorized users to view, add, edit, and delete documents within a user-defined folder hierarchy, as shown in the following figure.

![Document management](image)

Figure 9-10  Document management
New and changed documents can be made available immediately or can go through an approval process. PDM integrates with the same workflow engines as WPCP for document approval. In fact, PDM uses much of the same infrastructure as WPCP to support capabilities such as document versioning.

PDM uses Portal's access control capabilities to control which users can view or edit specific documents. Privileges can be set on a project, folder, subfolder, or individual document. Privileges set on a folder are inherited by all the documents and subfolders contained in that document, unless specifically blocked. This makes it easy to efficiently manage access rights on a large collection of documents.

PDM integrates with Portal search to allow users to search by document content, title, or description. Documents can be text documents such as HTML, or application-specific documents such as Microsoft Word. Word and many other file formats can be viewed using their corresponding viewers or optionally converted to HTML for users that do not have the necessary editors.

In addition, Portal Server includes Productivity Components, which may be accessed from PDM. Productivity Components function as embedded editors for rich text, spreadsheets, and presentations. These editors allow users to create and edit these document types from within their browser, as shown below.
PDM also supports a simple subscription capability. Users can specify folders or individual documents that they have an interest in. Whenever a document that a user is subscribed to changes, it is visible in the user’s Update folder for a specifiable period of time (1, 3, 7, or 14 days).

9.4.7 Search

WebSphere Portal provides integrated Web content search capabilities, including a search portlet, a crawler, a document indexer, and content categorization options. The search service can search local documents as well as Internet content. Portal search can index plain text documents as well as over 200 other file formats using built-in document filters.

The portal server’s built-in search engine is optimized for full-text searching of small and medium-sized collections where precision is essential. It efficiently applies state-of-the-art search algorithms producing high quality search results.
The search engine supports free-text queries, with query assistance and query word completion, and also supports wildcard and fielded search options. Search queries may also use advanced query operators (+ or -) to indicate keywords that must be in the document or keywords that must not be in the document. The search engine can search documents in any language, and also supports synonyms and stop word lists. Search results include document summarization, categorization, and search results clustering.

To prepare for searching, the search engine builds a full text index in order to search documents that are stored in the local file system. The indexer supports multi-word indexing for disambiguation and high precision. The index can be compressed, and the size can be controlled for situations where the size of the index needs to be limited. Administration portlets are provided for creating, updating, and managing the index parameters and optional categorization options. Content categorization options include classification to pre-established
categories, or a rules-based method to define content categorization filters to apply to indexed content.

![Image](image.png)

**Figure 9-13  Configuring the search index**

### 9.4.8 Federated search

WebSphere Portal also integrates several other search technologies. Portlets using IBM Lotus Extended Search and DB2 Information Integrator can access and aggregate other search engines and indexes in a distributed fashion. Customers seeking support for large document collections or support for searching a wide range of document types and data sources should consider using Lotus Extended Search (included with the WebSphere Portal Extend offering) or IBM’s Content Manager.

### 9.4.9 Extended search

Lotus Extended Search (ES) provides distributed, heterogeneous searching across Domino servers, databases, and the Internet, without the user having to know the details of these various systems. The result is single-point-of-access to a variety of data sources without requiring a new, central index. ES can search
and retrieve documents from repositories that include Lotus Notes 5.X and 6.x, Domino.doc, and Domino Server Domain Index.

ES also searches external sources such as Microsoft Index Server and Site Server, Microsoft SharePoint, Websphere Portal Search Engine Indexes, LDAP-compliant directories, 18 popular Web search sites and news sites, commercial content providers, and relational databases such as IBM DB2, Oracle, Sybase, Microsoft SQL Server, and other ODBC-compliant databases. Results can be ranked by relevancy over multiple data stores.

**9.4.10 Advanced search and document processing**

IBM Content Manager can manage data access across multiple sources such as content management repositories, e-mail systems, relational databases, file systems, Web sites (both intranet and Internet), and more. The developer’s interface for working with IBM Content Manager is called DB2 Information Integrator for Content. It integrates data sources across the enterprise through a unified set of APIs to simplify programming and speed development and deployment, while providing an interface layer that isolates portal applications from changes to underlying data repositories.

Documents can be full-text indexed/searched using the Information Integrator’s crawler and text search features. Many different document formats are supported, in addition to standard markup text such as HTML and XML. Documents can be categorized into taxonomies, enabling searches by category. APIs are provided for capturing and storing other metadata about documents.
Information Integrator has connectors for a variety of repositories, provided by IBM, Lotus, and other vendors such as Documentum and Filenet. Federated searches can be applied across multiple such repositories, and can exercise searching based on metadata, full text, and other specialized search properties such as Query by Image Content (QBIC), depending on the search services enabled for each repository.

The Text Analysis features support creating full-text indexes, and subsequent searching across all the text portions of the content sources configured for use in the portal. Sources can be accessed for indexing via the Web crawler, or via a metadata search. Portlets for accessing Information Integrator’s advanced and federated search functions are available from the portal catalog.

## 9.5 Security

With the explosive growth of business-to-business and business-to-consumer Web applications, e-businesses need to protect critical information assets from intruders and hackers. Service providers need similar protection when re-hosting e-business content and applications for their customers. Portal applications and resources are protected by:

- Managing user profiles (member services)
- Verifying user identity (authentication)
- Managing access to backend applications (single sign-on)
- Enforcing access policies (authorization)

This chapter highlights some of the security features that applications and portal administrators can exploit to better protect the portal’s valuable information assets.

### 9.5.1 Member services

Centralized administration of user identities, credentials, and permissions is desirable in many environments. The portal server includes facilities for defining portal users and managing user access rights.

The user and group subsystem includes Web pages where users can register and manage their own account information, administration portlets for managing user accounts and group information, plus a repository that stores all the information about portal users. It provides services to create, read, update, and delete users or groups in the repository. User profile information includes general information such as a user’s name and user ID, plus preference information such as news topics of interest, preferred language, etc. A user may be a member of one or more groups, and groups can contain other groups.
The default set of user profile attributes is based on the inetOrgPerson schema, which is supported by most LDAP directories. The user repository may consist of multiple data sources. By default, the repository consists of two data sources: It is a combination of a database and a directory server. The database may be any of the databases supported by WebSphere Portal. Any one of several LDAP directory products are supported, including the Netscape (iPlanet) Directory Server, Microsoft Active Directory, Novell eDirectory, Lotus Domino, and IBM Directory Server.

The mapping of user profile attributes to LDAP object classes is defined in the file wms.xml. This file specifies the names of the various data repositories and how they are navigated to retrieve user and group information. These settings are configured differently for each supported LDAP directory; if you want to try using a directory that is not supported, these values would need to be set appropriately for that directory server.

The file attributeMap.xml specifies the details of how each attribute is mapped to the LDAP directory or database. This mapping file also includes metadata about
each attribute such as its data type, whether it is required, whether it can have multiple values, etc.

9.5.2 Administration

Administration of users and groups can be performed by users themselves ("self care") or by portal administrators. The portal server includes forms for registering new users as well as administration portlets for updating user and group information.

```xml
<MemberServiceProperties>
  <Database ... DatasourceName="@DATA_SOURCE@" />
  ...
  <Directory userRDNname="uid"
    userMemberSubsystemAttributeName="logonId"
    userObjectClass="top/inetOrgPerson"
    ...
    orgRDName="dc"
    orgMemberSubsystemAttributeName="orgEntityName"
    orgObjectClass="top;domain"
    ...
    orgUnitRDNname="cn"
    orgUnitMemberSubsystemAttributeName="orgEntityName"
    orgUnitObjectClass="top;container"
    ...
    grpRDNName="cn"
    grpMemberSubsystemAttributeName="memberGroupName"
    grpObjectClass="top;groupOfUniqueNames"
    grpMembershipAttributeName="uniqueMember"
    ...
  </MemberServiceProperties>

Figure 9-16  Registration forms

The registration and self-care forms are easily modified to accommodate new attributes. You can simply add new data entry fields to the form, matching the field identifiers with the new attributes names. The enrollment servlet will automatically store the new data in the corresponding user attributes. The WebSphere Portal InfoCenter includes more information about customizing the implementation of the user repository, registration and self-care pages, and data validation classes.
9.5.3 Authentication

Authentication is the process of establishing a user’s identity. Usually, the portal server uses the authentication services provided by WebSphere Application Server. Another option is to use a third-party authentication server (such as Tivoli Access Manager WebSeal or Netegrity SiteMinder) that has a trusted association with the application server.

9.5.4 Identifying the user

A portal server uses form-based authentication. Form-based authentication means that a user is prompted through an HTML form for the user ID and password for authentication when trying to access the portal. The portal server requests the application server to validate the authentication information against a Lightweight Directory Access Protocol (LDAP) user registry.

WebSphere Application Server uses Lightweight Third Party Authentication (LTPA) as the authentication mechanism. A Common Object Request Broker Architecture (CORBA) Credential is used to represent authenticated users and their group memberships. When a user tries to access a protected resource, the application server intercepts the request and redirects the request to the login form. This form posts the user ID and password to the portal that requests the application server to authenticate the user. If the user can be authenticated, a valid CORBA credential is created and an LTPA cookie is stored on the user’s machine.

9.5.5 Third-party authentication servers

If your system uses another third-party authentication server, trust needs to be established between that proxy and WebSphere Application Server. This is done using a Trust Association Interceptor (TAI) module, which converts security information specific to the authentication proxy into a format that can be handled by the application server. The supported authentication mechanism depends on the capabilities of the third-party product.

When a user tries to access the portal, the third-party authentication proxy intercepts the request and challenges the user to authenticate. After a successful login, the original user request, along with additional security information in the request header, is passed to the application server. The format and content of this information is vendor specific. WebSphere Application Server uses the TAI module (that is specific to the third-party product) to extract the necessary security information from the request header.

TAI modules for IBM Tivoli Access Manager and Netegrity SiteMinder are packaged with the portal server (all editions). The WebSphere Application Server
InfoCenter includes information about creating custom TAI modules for other third-party reverse proxy servers.

### 9.5.6 Single sign-on

The portal server provides comprehensive single sign-on (SSO) support. Users want to be able to log on only once, and be known to the different parts of the portal server with the same consistent user credentials. Users should not be asked to do multiple logons simply because they access different portal applications.

The portal server supports single sign-on realms using WebSphere Application Server as well as authentication proxies. This means that the user needs to log on only once to gain access to all enterprise applications that are installed within the single sign-on realm.

The WebSphere Application Server uses Lightweight Third Party Authentication (LTPA) tokens to provide single sign-on. When a user is authenticated, the portal server creates an LTPA single sign-on cookie containing the authenticated user credential. This encrypted cookie conforms to the format used by WebSphere Application Server and can be decrypted by all application servers in the shared domain, provided they all have the same cipher key. This cookie enables all servers in the cluster to access the user's credentials without additional prompting, resulting in a seamless single sign-on experience for the user. To benefit from the LTPA method of single sign-on, the user's browser must support cookies and have its support for session cookies enabled.

### 9.5.7 Credential vault

Many portlets need to access remote applications that require some form of user authentication. For accessing applications outside the portal's realm, the portal server provides a credential vault service that portlets can use to store the user ID and password (or other credentials) for a user log into an application. Portlets can use these on behalf of the user to access remote systems. The credential vault supports either local database storage or IBM Tivoli's Access Manager for secure storage and retrieval of credentials.
Portlets obtain credentials by obtaining a CredentialVaultPortletService object and calling its getCredential method. With the returned credential, there are two options:

- Use passwords or keys from a passive credential, passing them in application-specific calls. Portlets that use passive credentials need to extract the secret out of the credential and do all the authentication communication with the backend application.

- Call the authenticate method of an active credential. Active credential objects hide the credential's secret from the portlet, with no way to extract it out of the credential. Active credentials provide additional methods to perform the authentication.

The latter case allows portlets to trigger authentication to remote servers using basic authorization, SSL client authentication, digest authentication, or LTPA without knowing the credential values. Using active credentials means that the portal authenticates on behalf of the portlet, and the portlet can simply use the open connection. While this may not be possible for all cases, it is the preferred technique.

For secure transmission of data, portlets can request a secure session (HTTPS) for accessing Web applications.
9.5.8 Persistent connections

Portlets that depend on remote connections require some way of maintaining that connection as users navigate through the portal. The portal provides a persistent backend connection service that maintains TCP/IP connections across page changes.

Some remote applications use forms-based logins and store cookies during the login form processing. The HttpFormBasedCredential can be used for handling these form-based logins and will store all the cookies that are returned as a result. For subsequent calls, the portlet can then ask the credential for an authenticated connection. This gives an HTTP connection with these cookies already set in the header. This way, portlets can maintain persistent, secure back-end connections.

9.5.9 Java security

The portal server implements the Java Authentication and Authorization Service (JAAS) architecture. JAAS provides a means for authenticating subjects and for providing fine-grained access control. JAAS is part of the standard Java security model; it gives applications independence from the underlying authentication and authorization mechanisms being used.

JAAS performs login and logout operations using a modular service provider interface. Credentials that are established through the portal server's JAAS login modules include CORBA credentials, user and group distinguished names, user ID and password, and LTPA tokens. In a distributed J2EE environment, portlets can use the JAAS API to access JAAS-enabled backend applications.

9.5.10 Authorization

After determining the user's identity, the portal server consults locally cached access control lists to determine which pages and portlets a user has permission to access.

The portal server enforces access control to portal assets, including portlets, pages, and user groups. The access control lists are stored in the portal's administration database. It is also possible to manage access control for specific resources in an external security manager, such IBM Tivoli Access Manager or Netegrity SiteMinder.

Access permissions are maintained using the Access Control administration portlet. Use this portlet to assign roles to individual users or to groups of users for specific portlets, pages, or documents. Roles are permission sets, such as the ability to view and update the corresponding item. Users may also delegate the
permissions they hold to other users. When a role is assigned to a user or a group on a container (such as a page that contains portlets or other pages, or a folder that contains other folders or documents), that role is inherited downward through the hierarchy unless it is specifically blocked. This makes managing access within a document library or an area of the portal easy.

Figure 9-18  Resource permissions
Figure 9-19  Managing access rights in the portal

Granting view access to a page or place means that other users will see pages and places when they log in. Granting view access to a portlet means that users can add it to their pages when they customize their portal experience. Granting edit access means that a user can set the portlet settings or change the contents of a page. Manage access means that a user can perform view and edit operations, and can delete the portlet or page.

9.5.11 Delegated administration

Granting view access to administration portlets is an effective way of delegating certain administrative tasks to other portal users. Those users can simply add the administration portlets to their personal pages, and then can perform whatever task the portlet is designed to do. This way, the user does not have to be given all administrative privileges or added to the portal administrator's group. Their administrative abilities are limited to only those tasks covered by the authorized portlets.
9.6 Personalizing the portal

Optimizing each user's experience in the portal is one of the goals of WebSphere Portal. To this end, the portal server provides end-user and administrative interfaces for customizing the content of portal pages, as well as the look and layout of the pages. With these tools, users can customize their own pages by selecting portlets and customizing the settings of each one. Users can also change the page layout and the color scheme (if the administrator has decided to allow this).

Figure 9-20  Portal page elements

9.6.1 Customizing pages

Users can have one or more personalized pages, navigating to each one from the home page. Pages are arranged in a hierarchical manner with any level of depth. Each page can have its own choice of color themes, skins, and page layouts. Themes are used to define the fonts, colors, spacing, and other visual elements; themes consist of cascading style sheets, JSP files, and images. Skins are decorations and controls placed around portlets, such as title bars, borders, shadows, etc. At each level of the page hierarchy, the lower pages can inherit the themes and skins from the page above them or can override one or both. Since the look and feel of each section can be completely different, sections of the site
can be used to create the appearance of different sites running on one portal server.

Each personalized page can have a different set of portlets. The portlets on a page can be selected by end users or by administrators, depending on their access rights for the page. Administrators can specify that certain portlets are required, so that end users cannot remove them or rearrange them. Pages can also be re-arranged to get a different navigation order for each user or group of users, when permitted by the administrator.

For end users, the portal provides a quick customizer interface for adding and re-arranging portlets. It is accessed by simply clicking the Edit Layout link at the top of the page you want to customize. Then it is just a matter of clicking the Add content button to add new portlets.

![Figure 9-21  Quick Customizer](image)

If you want to rearrange portlets on the page or delete portlets, do that ‘inline’ by toggling the page editing tools on and off and selecting the direction icons or delete icon associated with each portlet.
For administrators, there is also a more advanced page customizer. In the advanced customizer, the basic structure of a page is defined. Page layouts are fully dynamic, with the possibility of creating any arrangement of rows and columns. Columns and rows can be split to create new rows and columns. Columns can have a fixed size, specified in pixels or in a percentage of the overall page size, specified by using the width controls. To place portlets on the page, click the Add Content button and search for a portlet you wish to add to the page, and utilize page layout controls to customize the placement of portlets to pages.

9.6.2 Cascading portals

Many companies need to build portals that accommodate the needs of both a central organization as well as regional and local communities of users. For example, a company might want to have top-level design decisions made centrally, such as the visual elements and basic page layouts that reflect the corporate image and brand identity. Divisions or other organizational units might need the ability to add their own pages and content, or perhaps even sub-divide areas of the common pages.

WebSphere Portal supports the needs of companies who want to cascade portal definitions across the organization using the concept of derived pages.
Base pages are defined by top-level administrators who then permit subordinate administrators to further refine the page layouts and content to meet their individual needs. The refinement process can continue for any number of levels, until it finally reaches end users. If permitted, end users may customize their own pages by selecting and placing portlets and by changing portlet settings. In order to compute the page that a user finally sees, the portal server merges the page fragments defined by each successive refinement.

Administrators at different levels of the organization can lock the layout or content of any area of the page. For example, locking the placement of portlets means that users cannot move them or remove them from their pages. This way, a higher-level administrator can set up the basic structure of the page, and can fix certain portions or leave others open to modification by other administrators or by end users.

Lower levels may not override any restrictions imposed by higher levels. This means that a company can achieve both horizontal administration (across the company, business segments, locations, and users) and vertical administration, where administrators provide content that is dedicated for a specific page. Using page locks together with the portal's access control features, administrators can determine which content is:

- Mandatory: Users cannot delete the content from their pages.
- Recommended: The content is placed on the page initially, but users can delete it.
- Optional: The content is not placed on the page, but is available for end users to add it.
- Forbidden: Users cannot see this content at all.

### 9.6.3 Skins and themes

The portal server uses a system of JavaServer Pages templates, cascading style sheets, and images to define the look of the portal pages. You can modify these
to control any of the visual aspects of the portal, perhaps to add company-specific brand elements, or to achieve a different color scheme and visual style.

The system for defining color themes and portal skins has been enhanced to support multiple skins per theme; additional branding elements; navigation styles; and dynamic, browser-independent cascading style sheets.

![Figure 9-23 Setting themes and skins](image)

Skins and themes can now be applied to each page rather than being restricted to having only one for the overall portal. Different skins can be applied individually to portlets, so that the portal look can be fine-tuned to meet any needs.

![Figure 9-24 Applying skins and themes](image)
9.6.4 Brand elements

All of the visual elements of the portal, including the masthead, the navigation areas, graphics, portlet title areas, and stylesheets, can be changed to give the portal a custom look.

Standard file formats, such as JPEG, GIF, CSS, and JSP files are used for defining the look and the layout of the portal. If you examine the structure of the portal server's installation folder, you will notice folders named skins and themes, with folders html, wml, and chtml beneath them.

These folders contain most of the files used for defining the basic structure of the portal's home page, its color schemes, and portlet decorations. You can make copies of these folders and modify the contents to achieve the visual image that your company needs. The theme administration portlet registers the new files.
9.6.5 Navigation

The portal's navigation capabilities have been improved significantly. Now, instead of a simple page list for each place, it is possible to create very complex navigation trees, with both page links and links to external URLs shown in the navigation area. You can create labels that logically group certain navigation elements, so with more links indented beneath these labels.

![Page navigation administration](image)

*Figure 9-26  Page navigation administration*
9.6.6 Virtual portals

Using different themes for each place, a single installation of the portal server can give the appearance of supporting many “virtual” portals. For example, a company might want to have a different portal for each division, or it might want to have special branding of its business-to-business portal for each business partner. Each virtual portal can be designed using one or more places, each with its own theme, skins, page layouts, and access permissions. Administration of the associated portal resources can be delegated to a different administrator.

9.6.7 Universal access

Earlier, we hinted at the portal’s support for international use. The entire system of page templates, themes, skins, and portlet rendering is fully enabled for internationalization (including double byte and bi-directional languages) and for accessibility to people with disabilities. The portal server generates markup that complies with the American Disability Act (ADA) as defined in Section 508 Web Accessibility Standards and meets the guidelines of the W3C Web Accessibility Initiative.

For globally accessible portals, the portal server will search for and select the proper JSP pages based on the target browser, and target browser’s settings for language and country. To enable a portlet for language selection, its JSP files are packaged using the following directory structure:

```
WAR / markup_type / language / country / variant /jspname.jsp
```
The portal server searches for the JSP for a portal from the most specific to the least specific, as in the following example.

2. /html/en/US/portletJSP.jsp
3. /html/en/portletJSP.jsp
4. /html/portletJSP.jsp
5. /en/US/IE/portletJSP.jsp
6. /en/US/portletJSP.jsp
7. /en/portletJSP.jsp
8. /portletJSP.jsp

Another good technique for handling translation issues is to use the portal server’s JSP tag library. It includes a tag for identifying and separating translatable text, so that you do not have to create and maintain separate JSP files for each language. Using this technique, you could create one JSP view, tagged like this:

```html
<H2><wps:text key="heading.hello" bundle="mystrings"/></H2>
```

When this page is processed, the strings are substituted using standard Java resource bundles.
For portals that need to support many languages or for portals that include rapidly changing portal content, WebSphere Translation Server offers automatic translation technology. This technology is useful for either real-time or offline translation of human languages.

WebSphere Translation Server provides quick, inexpensive, convenient "gist" text translations whenever a professional translation is not feasible because of availability, time, or cost. The dictionaries can be tuned so that idiomatic expressions and specialized terminology are interpreted correctly. It supports bi-directional translation of content for English to and from French, Italian, German, Spanish, Japanese, and Chinese (simplified and traditional). Unidirectional translation of content from English to Korean and Brazilian Portuguese is also supported.

### 9.6.8 Personalization

The WebSphere Portal offerings include the WebSphere Personalization server. The purpose of the personalization server is to allow you to target content to specific users in support of your portal's business goals.
WebSphere Personalization provides facilities that allow subject matter experts to select content suited to the unique needs and interests of each site visitor. The Web-based tools help companies quickly and easily leverage content created by line of business and subject matter experts.

A personalization solution involves three basic components:

- **User Profile**: Information about users of the site, including attributes about the users
- **Content Model**: Defines attributes about the content, such as product descriptions, articles, and other information
- **Matching Technology**: Engines that match users to the right content; includes filtering, rules, recommendation engines, or combinations of all three

The WebSphere Personalization server and WebSphere Portal server share a common user profile and a common content model. The model is based on the WebSphere resource framework interfaces classes. This means that personalization rules can easily be added to portlets to select portal content and target it to the portal’s registered users.

The basic steps involved in personalization involve classifying site visitors into segments, and then targeting relevant content to each segment. Business experts create the rules for classifying users and selecting content, using Web-based tools.
WebSphere Personalization also includes a recommendation engine, which provides collaborative filtering capabilities. Collaborative filtering uses statistical techniques to identify groups of users with similar interests or behaviors. Inferences can be made about what a particular user might be interested in, based on the interests of the other members of the group.

Also included with WebSphere Personalization are new campaign management tools. Campaigns are sets of business rules that work together to accomplish a business objective. For example, an HR manager may want to run a campaign to encourage employees to enroll in a stock purchase plan. The HR manager would define a set of rules as shown to accomplish this business objective. Campaigns have start and stop dates and times and can be e-mail and Web page based. Several campaigns can run simultaneously and they can be prioritized.
Implicit profiling services can collect real-time information about site visitor actions and then construct personalization business rules using this data. Implicit profiling tracks the areas of a site where a user is most active in order to determine the user's interests. For example, if a user clicks on articles about football, it is possible to determine that he is interested in sports, even if he did not explicitly indicate this interest when registering at the portal.

To analyze the effectiveness of the site and its personalization strategies, the server provides logs that can be analyzed by IBM Tivoli Web Site Analyzer or WPCP’s Reporting capabilities. Web Site Analyzer can then create reports for the portal's business owner. This helps the company measure the effectiveness of the business rules and campaigns in achieving its objectives.
9.7 Administration

Administration of the portal is done through the portal itself, either in a centralized or delegated fashion. Administrators can deliver a new service to users simply by adding new portlets to the pages of the portal. Since these are portlets, just like bookmarks or reminders or news or any other portlets, administrators can control access to them, place them on portal pages, and perform any of the usual steps.

Administrative portlets are provided for adding portlets to the portal’s registry; managing users, groups, and access control lists; clipping Web pages;
publishing Web services; setting portal-wide settings; managing logs; and other common tasks. In this chapter, we describe some of the administration portlets and describe what they do for you. You have already seen some of these in previous chapters, so we just focus on the ones that have not been covered already.

9.7.1 Portal settings

In the global settings portlet, administrators can change portlet settings such as the default language, the cache timeout values, etc. In addition, there are settings that control how new user sessions are handled, and what to do when a user tries to access a portlet without authorization. Unauthorized access can be ignored (in other words, the portlet is not displayed), or the portlet can be replaced by an informative message so that the user can take the necessary actions to correct the situation. Returning users may wish to pick up where they previously left off, so there is a setting to retain the state of the last visit and return to that page the next time.

Figure 9-32  Global portal settings

9.7.2 Web clipping portlet

One of the most important portlets is the Web clipping portlet. This portlet is used to display sections of existing Web pages. You can visually select portions of the
page or clip all the text between specific tags. This way, you can precisely control what markup is extracted. The portlet can optionally rewrite the links inside the clipped page, which is useful for displaying existing pages without leaving the portal's navigation structure. Each time you clip a Web page, a new portlet is created in the portal's registry. Whenever the new portlet is displayed, it retrieves the current version of the Web page and extracts the clipped portion to display.

![Web Clipping Editor](image)

*Figure 9-33  Clipping Web pages*

Some sites that you clip might require authentication. The clipping portlet provides options for no security, basic authentication, or form-based authentication. The credentials can be provided by the user, or filled in by the administrator.

### 9.7.3 Managing portlets

When installing new portlets, you can use Web archive files (WAR) from your local file system, or you can install portlets that have been previously published in a UDDI directory.
The new portlet is automatically activated, but with no special permissions. Access control for the new portlet is inherited from the default portlet settings, but can be set explicitly by the administrator.
9.7.4 Managing the portlet catalog

Once the portlet is installed, you can copy it, set its configuration parameters, activate or deactivate it, or uninstall the portlet.

![Managing the portlet catalog](image)

Figure 9-36 Managing the portlet catalog

9.7.5 Users and groups

In previous versions of WebSphere Portal, it was necessary to use your LDAP directory's administration tools to manage user and group information. Now you can use the portlets provided to manage user and group information without leaving the portal. You can also manage a user's group memberships. These portlets provide search capabilities and pagination to allow the administrator's interface to scale and manage a large number of users and groups.
The portal server uses group membership information to determine what pages, portlets, and documents a user is authorized to view and edit. Users can be members of one or more groups, and groups may contain other groups. Users will be allowed access to portal resources when access is granted to any group that the user belongs to. Access rights can also be granted to specific individuals, but most companies find that it is easier to manage the access rights of groups instead.

9.7.6 Web services

A Web service is an interface that describes a collection of network-accessible operations. The interface is described using a standard XML description language called Web Service Description Language (WSDL), so that the service can be invoked without prior knowledge of the platform, language, or implementation design of the Web service. Web services are located using the standard Universal Description and Discovery Interface (UDDI), which may be a private or public registry of Web services.
WebSphere Portal provides extensive support for Web services. Portal administrators can publish and bind remote portlets as Web services, making the remote portlets available in the portal's registry dynamically. When a remote portlet is used, its services are invoked using Simple Object Access Protocol (SOAP) or other transport protocols.

Note that in WebSphere Portal Version 4.2, this was done using a proprietary remote portlet protocol. Since that time IBM has lead a standardization effort that lead to the definition of WSRP. The V4.2 protocol was not shipped in WebSphere Portal Version 5, and will incorporate support in the WebSphere Portal Version 5 platform for the WSRP standard version soon after the standard is finalized.

### 9.7.7 Portal analysis and logging

Administrators can control the tracing and logging activity and monitor user activity through the Portal Analysis administrative options and portlets that enable tracing and track frequent user information, and also by modifying the configuration properties files of the logging subsystem.

The portal server also records user activity in logs that can be processed by Tivoli Web Site Analyzer. Overall usage statistics such as logins and logouts are tracked, along with portlet and page usage statistics.

| Table 9-6  Portal usage reports from Tivoli Web Site Analyzer |
|-------------|-----------------------------------------------------------|
| **Project Summary** | Displays summary statistics about the analysis, such as the total number of hits, sessions, visitors, and page views |
| **User Agent Ranking** | Displays a ranking of user agents used by visitors to your site |
| **Browser Ranking** | Displays a ranking of browsers used by visitors to your site |
| **Platform Ranking** | Displays a ranking of users to your site by frequency of visits |
| **User Ranking** | Displays a ranking of users to your site by frequency of visits |
| **Page Ranking** | Displays a ranking of the pages viewed or edited by visitors to your site |
| **Portlet Ranking** | Displays a ranking of the portlets viewed by visitors to your site |
9.8 Collaboration

Corporate portals connect people to the applications, content, and resources they need. Portals also connect people to each other, through community pages, shared bookmarks, contact lists, and personal productivity portlets.

Collaboration is really about people working efficiently and effectively in teams to solve business problems. The portal server includes portlets and services designed to support the team and its activities, with messaging, document libraries, user profiles, inboxes and calendars, online directory services, team workplaces, and electronic meetings. Users can access these collaborative services in the context of what they are currently doing, instead of having to leave the portal to open another application. For example, while working in the portal, users can easily see who else is online and then send an instant message, send an e-mail, or add a person to their contact list. Collaborative portlets have advanced built-in features that allow portal users to take actions on documents or user names that appear in a portlet. Directly from the portlet, a portal user can see if other users are online, and select from a menu of options to interact with another user.

9.8.1 Collaborative portlets

The collaborative portlets provide access to a variety of collaborative applications that use Lotus Notes databases hosted on Domino servers: The Notes and Domino portlet offers Mail, Calendar, To Do, Notes View, TeamRoom, and Discussion; while the Domino Web Access (iNotes) portlet offers Mail, Calendar, To Do, Contacts, and Notebook environment.
There are also portlets for Domino.doc, Discovery Server, Lotus Team Workplaces (QuickPlace), and Lotus Instant Messaging and Web Conferencing (Sametime).

**9.8.2 Collaboration Center**

Collaboration Center offers an integrated framework of e-workplace components for finding, connecting to, and working with people. These include the People Finder, the Web Conference center, Team Workspaces, and Sametime portlets. Using these components, users have immediate access to a searchable directory of people that is integrated with their workplaces and their e-meetings within the collaborative portal. Users can find people in the directory, see their online status, and interact with them using instant messaging and other actions provided by people links. In addition to search features, the People Finder portlet provides views of each person's directory record and his or her place in the organizational context.
Table 9-7 summarizes the collaborative portlets.

**Table 9-7  Collaborative portlets**

<table>
<thead>
<tr>
<th><strong>Domino Web Access (iNotes)</strong></th>
<th>Provides access to a Lotus iNotes server, Mail, Calendar, To Do List, Contacts, Notebook functions.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lotus Notes and Domino</strong></td>
<td>Mail, Calendar, To Do, Notes View, TeamRoom, and Discussion</td>
</tr>
<tr>
<td><strong>Domino.Doc</strong></td>
<td>Displays a Domino.Doc library inside the portlet.</td>
</tr>
<tr>
<td><strong>Lotus Team Workplaces</strong></td>
<td>Displays a Lotus QuickPlace component inside the portlet.</td>
</tr>
<tr>
<td><strong>(QuickPlace)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Lotus Instant Messaging</strong></td>
<td>Launches a Sametime Java connect client from within the portal.</td>
</tr>
<tr>
<td><strong>(Sametime Chat)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Web Page</strong></td>
<td>The Web Page portlet will show the contents of an Internet or intranet Web page in a scrollable portlet.</td>
</tr>
</tbody>
</table>
## 9.8.3 Related products

WebSphere Portal Extend integrates tightly with Lotus' world-class collaboration and knowledge management products, including Lotus Team Workplaces (QuickPlace), Lotus Instant Messaging and Web Conferencing (Sametime), and Lotus Discovery Server, by providing portlets and services to access these products seamlessly from the portal.

<table>
<thead>
<tr>
<th>Product</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovery Server</td>
<td>Searches knowledge map categories and content or displays a knowledge map.</td>
</tr>
<tr>
<td>People Finder - a component of Collaboration Center</td>
<td>Locates a person by name or profile characteristics.</td>
</tr>
<tr>
<td>My Team Workplace - a component of Collaboration Center</td>
<td>Shows, searches, and creates QuickPlaces.</td>
</tr>
<tr>
<td>Web Conferencing - a component of Collaboration Center</td>
<td>Schedules, finds, and attends e-meetings.</td>
</tr>
<tr>
<td>Sametime Who Is Here</td>
<td>Displays a list of names for chat, modifies groups for display and status message.</td>
</tr>
<tr>
<td>Sametime Contact List</td>
<td>Displays a list of people who are online and available for chat, either in the current page, or a named virtual page.</td>
</tr>
</tbody>
</table>

## 9.8.4 Lotus Team Workplaces (QuickPlace)

Lotus QuickPlace provides workspaces for sharing and organizing ideas, content, and tasks. It includes tools to manage team projects and schedules, organize discussion threads, and share documents.

The Lotus QuickPlace user interface is shown inside a portlet. The integration between the portal and the QuickPlace includes single sign-on, so that users who access the portal can access the Lotus QuickPlace seamlessly, without requiring an additional login.
9.8.5 Lotus Instant Messaging and Web Conferencing (Sametime)

Lotus Sametime provides instant messaging, shared white boards, and application sharing for electronic meetings. Sametime functionality is integrated into the portal for access to chat sessions and buddy lists, as well as people and place awareness. Awareness is the ability to tell who the place members are and to find out whether they are online, offline, or not available.

Sametime provides other services that can also be integrated through portlets: Application sharing, white boarding, and online meetings.

9.8.6 Lotus Discovery Server

Lotus Discovery Server is a separately purchased product that creates expertise and knowledge maps by analyzing and categorizing documents. It creates profiles of users based on their document activity, including their topics of interest and their area of expertise.
Discovery Server also examines user activity such as reading documents, responding to documents, timeliness of interactions, or links to specific documents. This way, Discovery Server can determine the relative proficiency of individuals to content categories. These proficiency indicators are called affinities, and they indicate the relative expertise of individuals to particular business areas of the organization.

The Discovery Server continually assesses the strength of affinities using metrics, ensuring that individuals with recent and high quality expertise ratings are found and presented to users seeking expertise when browsing the Discovery Server Knowledge Map interface.

Lotus Discovery Server is purchased separately from WebSphere Portal.

### 9.8.7 Collaborative services

Lotus Collaborative Components are building blocks (APIs and JSP tag libraries) for integrating the functionality of Lotus Domino, Lotus Sametime, Lotus QuickPlace, and Lotus Discovery Server into the portal. Developers can leverage the features of Lotus Domino, Lotus QuickPlace, Lotus Sametime, and Lotus Discovery Server by using these components to add user interface extensions to their portlets and portal pages.

The collaborative services hide the configuration details of the Lotus products installed within an enterprise. Instead of working with more complex product APIs, Lotus collaborative components provide developers with an easier method of integrating core collaborative features into any portal or portlet.

The benefit of using Collaborative Components is that they provide standardized access to Lotus applications, with easier APIs that are optimized for the portal. Security context is handled automatically, and upgrades of the backend systems are transparent to the portlets. The Collaborative Components work across versions of Domino 4.67 and later.

<table>
<thead>
<tr>
<th>Table 9-8  Lotus Collaborative services</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>People Service</strong></td>
</tr>
<tr>
<td><strong>Domino Data Service</strong></td>
</tr>
</tbody>
</table>
The People Service provides an awareness function that is available through a custom JavaServer Pages tag. Adding this custom tag creates an online status indicator and a pop-up menu of action choices that access various collaborative services. For example, you could start with a portlet that produces markup like that shown below.

```
<%@page language="java" import="com.lotus.cs.*, java.util.*" %>
<%@taglib uri="/WEB-INF/tld/people.tld" prefix="peopleservice" %>
<%@taglib uri="/WEB-INF/tld/portlet.tld" prefix="portletAPI" %>
<portletAPI:init/>
<h1>People Awareness Example</h1>
<% 
CSEnvironment environ = CSEnvironment.getEnvironment(request);
CSCredentials cred = environ.getCredentials(request);
%>
<H2>
<peopleservice:person>
<%=cred.getUserLoginName()%>
</peopleservice:person>
</H2>
```

*Figure 9-41  Markup*

The lines highlighted in blue are the special tags needed to make the online presence indicator work. The portal user might see something similar to Figure 9-42.
A great way to apply this idea is by adding the people awareness tags to a portlet that accesses an enterprise application, such as a PeopleSoft or Siebel function.

The exact menu items depend on details about the specific place, the permission level of the current user, and what collaborative products and services are installed in the portal environment. All of the collaborative services except for the PeopleService tags can be used in both desktop and mobile portlets.

9.9 Application integration

By definition, a portal provides access to content, data, and services located throughout the enterprise. These include not only predefined connectors and portlets, but also additional connectors and portlets created by various tools.

Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems are excellent candidates for portlets because efficient, personalized access to these functions provides measurable return on your portal investment. WebSphere Portal includes portlets that help you access a variety of ERP and CRM systems.

9.9.1 Enterprise portlets

The WebSphere Portal Application Integrator (WPAI) is a business tool that provides a fast way to build new portlets that access various kinds of Enterprise Application systems, including SAP, PeopleSoft, Siebel, WebSphere Portal Content Publisher, and Domino applications. It also works for relational databases, such as IBM DB2, Oracle, Informix, Microsoft SQL Server, and others.

The goals of the integrator framework were to provide a simple way for portlet administrators to build new portlets. Simply put, each tool is a portlet that builds new portlets for a specific backend system using a set of predefined interaction models. Operating the builder tool is simple and easy for administrators or business users. It works by querying the backend system for metadata that describes the business object that the new portlet will work against.
Once the business object is selected, the person building the portlet selects which fields and operations she wants to enable in the portlet. The operations include searching, updating, deleting, and creating new records.
Once the selections are made, the portal server stores the configuration information needed to run the portlet. There is no new Java code or service that needs to be deployed, just the configuration information for each portlet.

Using this approach, anyone who understands the usage pattern of the enterprise application system can build a portlet in just a few minutes. In the future, it may be possible to build additional connectors for other applications such as Oracle Applications, Ariba, Documentum, and others.

9.9.2 Commerce

Many portals need to include access to commerce functions, such as portlets that access a catalog of products and services, or portlets that present a catalog of products and services, prices, discounts, accept orders, process requests for quotes (RFQ), or provide contract-driven views into the catalog to give business customers personalized service.

Portals may provide these functions to consumers or to business partners, for procurement or electronic exchange that links buyers and sellers of goods.

IBM WebSphere Commerce enables an extended enterprise; a network of employees, suppliers, and partners collaborating to meet the full spectrum of customer needs by including configurable business processes to speed time to value for customer and partner facing projects. The WebSphere Commerce business context engine controls these business processes to present information and functionality to your constituents in a context-sensitive manner. For example, partners will have access only to product information that they are qualified to resell, with parts, inventory information, site language, and currency specific to their geography.
WebSphere Commerce Portal combines WebSphere Portal V5 and WebSphere Commerce V5.5 software solutions in a single package to deliver the next generation e-business infrastructure. With portal as your single point of user interaction your constituents are now presented with a context-sensitive and targeted collection of content, applications, resources, and processes to create commerce-enabled portals that provide consolidated, personalized access to commerce functions via the Web or wireless devices. This capability is available through service offerings from IBM and business partners.

### 9.10 Mobile portals

Next generation portals will be accessible through more than just traditional desktop browsers. Access through handheld or mobile devices is becoming increasingly important. The portal server currently supports mobile devices by generating portal pages in three markup languages: HTML for desktop computers and some personal digital assistants; WML for WAP devices, which are typically mobile phones; and cHTML for mobile devices in the NTT DoCoMo iMode network. Users can customize a unique home page for each device, selecting the content and applications that are most useful on the device.

When the home page is requested, the page is produced by first detecting the type of device that is making the request, and then assembling the portlets, which each render their contents in the appropriate markup language. When a user customizes the home page for a particular device, the portlet selection list only shows portlets that can actually produce markup appropriate for that device.

Thus, the list of available portlets for each device depends on what the portlets can actually do.

Some portlets may be available for all the supported devices, while others may be available only on a single device. The user interface design of each portlet varies from device to device, so that the user’s experience can be fine-tuned. Thus, the user’s home page and each of the portlets might be very different on a mobile phone versus a desktop browser.
9.10.1 Supporting new devices

The portal server supports several different markup languages so that portlets can render themselves for a variety of desktop and mobile browsers. Portlets that do not natively support the device markup can optionally be transformed using Transcoding technology. This means that portlets can easily and automatically support mobile devices, even if the portlet developer did not explicitly support that device.

9.10.2 Clients and markups

The portal server's page aggregation subsystem supports several markup languages and recognizes certain browsers and mobile device user agent signatures, out of the box. The framework for supporting markup languages is open and extensible, so it is easy to support additional markups or new devices.

To support new browsers and devices, you add new markup and clients using the corresponding administration portlets. In the markups portlet, the markup name indicates the name of the folders that are used to store the page templates and the theme or skin files matching that markup language.
To add a markup, create a new entry specifying the MIME type and the character set associated with that markup. You also need to add all the JSP templates associated with supporting a markup, such as new layouts, screens, skins, and style sheets.

When the portal server receives an HTTP request, it matches the values in the user agent header against known patterns that identify common browsers for desktops, mobile phones, and other devices. Entries for common clients are already set up, but you can add new ones using the Manage Clients portlet.
Figure 9-48  Enabling the portal for new client browsers

The ordering of the client entries determines the order in which the patterns are applied against incoming HTTP requests. For further information, see the WebSphere Portal InfoCenter.
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 167. Note that some of the documents referenced here may be available in softcopy only.

- WebSphere Portal on z/OS, SG24-6992
- WebSphere Commerce V5.5 Handbook, Customization and Deployment Guide, SG24-6969

Other publications

These publications are also relevant as further information sources:

- Fundamentals Guide, IBM WebSphere Commerce V5.5, available at:
- Rapid Java and J2EE development with IBM WebSphere Studio and IBM Rational Developer at:

Online resources

These Web sites and URLs are also relevant as further information sources:

- Rational Application Developer:
  http://www.ibm.com/software/awdtools/developer/application
- Rational Web Developer home page
  http://www.ibm.com/software/awdtools/developer/web
- J2EE Connector Architecture (JCA) 1.0 Specification
WebSphere Product Family Overview and Architecture

- IBM e-business on demand Web site
  http://www.ibm.com/ondemand

- WebSphere Application Server - Express requirements
  http://www.ibm.com/software/webservers/appserv/express/requirements/

- WebSphere Application Server requirements
  http://www.ibm.com/software/webservers/appserv/was/requirements/

- WebSphere Application Server Network Deployment requirements
  http://www.ibm.com/software/webservers/appserv/was/network/requirements/

- WebSphere Application Server Enterprise requirements
  http://www.ibm.com/software/webservers/appserv/enterprise/requirements/

- WebSphere Application Server for z/OS requirements
  http://www.ibm.com/software/webservers/appserv/zos_os390/requirements/

- WebSphere Application Server for iSeries requirements

- WebSphere Application Server Network Deployment for iSeries requirements

- WebSphere Application Server - Express for iSeries requirements

- WebSphere Portal

- Portlet API documentation
  http://www7b.software.ibm.com/wsdd/zones/portal/portlet/5.0api/WPS/

- WebSphere Portal home page

- WebSphere Portal requirements

- IBM WebSphere Portal Enable for iSeries
  http://www.ibm.com/software/genservers/portaliseries/

- WebSphere Portal for z/OS and OS/390 V4.1 home page
  http://www.ibm.com/software/genservers/portalzos/
> WebSphere Business Integration Modeler home page
  [link](http://www.ibm.com/software/integration/wbimodeler/)

> WebSphere Business Integration Workbench requirements
  [link](http://www.ibm.com/software/integration/wbimodeler/workbench/requirements/)

> WebSphere Business Integration Workbench Server system requirements
  [link](http://www.ibm.com/software/integration/wbimodeler/server/requirements/)

> WebSphere Business Integration Serversystem requirements
  [link](http://www.ibm.com/software/integration/wbiserver/requirements/)

> WebSphere InterChange Server requirements
  [link](http://www.ibm.com/software/integration/wbiserver/ics/requirements/)

> WebSphere Business Integration Message Broker requirements
  [link](http://www.ibm.com/software/integration/wbimessagebroker/requirements/)

> WebSphere Business Integration Adapters home page
  [link](http://www.ibm.com/software/integration/wbiadapters/)

> WebSphere Business Integration Toolset home page
  [link](http://www.ibm.com/software/integration/wbitools/)

> WebSphere Business Connection requirements
  [link](http://www.ibm.com/software/integration/busconn/requirements/)

> WebSphere Data Interchange Client requirements
  [link](http://www.ibm.com/software/integration/wdi/requirements/)

> WebSphere Business Integration Monitor Web-Client requirements
  [link](http://www.ibm.com/software/integration/wbimonitor/requirements/)

> WebSphere MQ 5.3 requirements
  [link](http://www.ibm.com/software/integration/websphere/mqplatforms/supported.html)

> Portal catalog download site
  [link](http://www.ibm.com/software/genservers/portal/portlet/catalog)

### How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)
Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services
Index

Numerics
3270 cclxx, cclxxii
5250-based applications xliii

A
Access beans cccxlivii
access control 10
Access Control administration portlet 127
access intent xlix, cccxxvii
access intent policy cccxxvii–ccxxviii
Activate At policy ccli
activation.jar cxiv, cxciii
ActivationSpec clxxxii
Active Script debugger ccclv
activities
Process Choreographer cccxiii
ActivationSpec clxiv
activity sessions and last participant support I
ActivitySession
beginSession() method ccl
Client scoped life cycle cclii
Client side demarcation cclii
Coordination of one-phase commit resources cclii
endSession() method ccl
Option A cclii
Option B cclii
Option C cclii
Option C+ cclii
resetSession() method ccl
Usage Scenarios cclii
acwa.jar cclxxi
Admin service lxxvii, xciv
admin_host xcv, clxix
administrative console cxviii, cc
WebSphere Application Server cclxivii
administrative service clxi, clxv
aggregation cclxii
Alarm cclxvi
AlarmManager cclxiv–cclxvi
Apache Tomcat cccxx, cccxxiv, cccxlxi, cclxiv
application
shutdown cclxxii

B
back office assets 40
background processing cclxii
Base rule cclxii
batching cxlvii
bean cache policy ccl
Bean Scripting Framework (BSF) cxix, cci
bean-managed transaction lxxvii, clxiii
bindings cccxxxiii
configured xcii, clxvii

start up cclxxi
application callable receive cclix
application client xxx
Application client container lxxv
application client container clxv, clx
application client module cclxv
application profiling xlix, cccxxviii
API cclxvi
decision algorithm cccxxviii
Application Response Time (ARM) agents xciii, clxxiv
application server lxxvi, cxx, clvi, ccl
clustering lxxvi, clvi
authentication cclxxii
collision cclxxii
definition lxx
Application Server Toolkit xxx, ccxviii, ccx
application service levels cccxxviii
Aptrix 112
ARIS to WebSphere MQ Workflow Bridge lxii
Assign activity cclxiv
asynchronous beans lii, cclxi, cclxii, cclxv
AsynchScope cclxiv–cclxv
authentication cvi, clxxxii, cclxxvi, clxxxv, 10, 124
authorization cvii, clxxxii, 127
automation cclxxiv
autonomic manager cclxxiv, cclxxvi, cclxxvii
average drop rate cclxxv
average queue length cclxxvi
average queue wait time cclxxv
average response time cclxxvi, cclxxv, cclxxvii
average service time cclxxvi
average throughput cclxxv

© Copyright IBM Corp. 2004, 2005. All rights reserved.
name bindings xci, clxvi
BPE container ccxiv
broker ccxc, ccxciv–ccxcv
broker administration cccix
broker archive file (BAR file) ccxci, cccxi
broker domain ccxci, ccxcii–ccxciv, cccx
brokered cooperation 108
builder cccxii
Built-in Mail Provider cxiii, cxciii
bus member clxxix
business analyst ccliv
Business Integration perspective ccxvi
business integration tools cccxix
Business Interaction Engine 2
Business Object Designer lxi
business policies 52
Business process
generate deploy code ccxxvii
test with Web Client ccxxviii
Business Process Execution (BPE) container ccxiv
Business Process Execution Language for Web
Services (BPEL4WS) ccxv
Business Process Web Client ccxxviii
Business Rule Beans
Base rule cclvii
Non-classifier rule cclvii
business rule beans li, cclv
classifier rule cclv
client component cclv
EJBs cclv
framework ccliii
persistence cclv
rule cclv
RuleImplementor cclv, cclv
trigger point cclv
TriggerPoint object cclv
XML cclv
business rules ccliii

catalog entries 12
Catalog subsystem 12
catalogs 52
cell
definition lxxx, clv
CICS xli–xlii, xcix, clxxiv, cccxvii–cccxxvii, cclxx,
cclxxii, cclxxxv iv
Class Diagram Editor cccxlviii
Class File Editor cccxi
classifier rule cclv
click-to-action portlets 108
client application container lxxvii
client caching clxix
client/server topology clxix
CLISTs xlii, cccxliv
cluster lxxxii–lxxxiii, cclvi, clv–cclv, clxxix,
cclxxi–cclxxii, cccvi–cxcvii, cvi
definition lxxii, clv
workload management lxxxiv, cccxviii
CMM cclv
CMP over Anything I
CMR cccxxviii
COBOL cccxiv, cccxxv–cccxxvii, cclxx–cclxxvii
collaboration 150
Collaboration Center 100–101, 151
Collaborative Components 99–100, 155
collectives cccvii
command registry 51
Common Secure Interoperability civ, clxxv
Compact HyperText Markup Language (CHTML)
cccxxviii
compensation support ccclix
compiled language debugger ccclix
component test tools ccclix
concurrency limits cccxvi
concurrent requests cccxix
Concurrent Versions System (CVS) cccxxiii, ccclix
configuration data 51
Configuration Manager ccxii, cccxiii–ccxv,
cccx–cccxi, 14
configuration repository cxix, cci–ccii
Connection Manager lxxvii, cclii
Container Managed Messaging cclv
Container Managed Relationship ccxxxvii
container-managed transaction lxxviii, ccliii
Content Publisher 99
content-based filtering cccv
content-based message routing cclxxxviii–cclxxix

cache disk offload lxxxix, clxiv
cache replication lxxxix, clxiii
caching clxviii
caching batching clxviii
Caching Proxy xxix, ccx
caller identity cccxix
campaigns 52
Cascading Style Sheet cccxiv, cccxxxiii

catalog entries 12
Catalog subsystem 12
catalogs 52
cell
definition lxxx, clv
CICS xli–xlii, xcix, clxxiv, cccxvii–cccxxvii, cclxx,
cclxxii, cclxxxv iv
Class Diagram Editor cccxlviii
Class File Editor cccxi
classifier rule cclv
click-to-action portlets 108
client application container lxxvii
client caching clxix
client/server topology clxix
CLISTs xlii, cccxliv
cluster lxxxii–lxxxiii, cclvi, clv–cclv, clxxix,
cclxxi–cclxxii, cccvi–cxcvii, cvi
definition lxxii, clv
workload management lxxxiv, cccxviii
CMM cclv
CMP over Anything I
CMR cccxxviii
COBOL cccxiv, cccxxv–cccxxvii, cclxx–cclxxvii
collaboration 150
Collaboration Center 100–101, 151
Collaborative Components 99–100, 155
collectives cccvii
command registry 51
Common Secure Interoperability civ, clxxv
Compact HyperText Markup Language (CHTML)
cccxxviii
compensation support ccclix
compiled language debugger ccclix
component test tools ccclix
concurrency limits cccxvi
concurrent requests cccxix
Concurrent Versions System (CVS) cccxxiii, ccclix
configuration data 51
Configuration Manager ccxii, cccxiii–ccxv,
cccx–cccxi, 14
configuration repository cxix, cci–ccii
Connection Manager lxxvii, cclii
Container Managed Messaging cclv
Container Managed Relationship ccxxxvii
container-managed transaction lxxviii, ccliii
Content Publisher 99
content-based filtering cccv
content-based message routing cclxxxviii–cclxxix

cache disk offload lxxxix, clxiv
cache replication lxxxix, clxiii
caching clxviii
caching batching clxviii
Caching Proxy xxix, ccx
caller identity cccxix
campaigns 52
Cascading Style Sheet cccxiv, cccxxxiii
contextual information  cclxxix
contracts  52
cookies  xcv–xcvi, clxx, 12, 125
CORBA  xlv, xlvii, liv, xci, clxv, 124, 127
corbaloc  xci, clxvii
corbaname  xci, clxvii
core data  51
core group  ccxiv
correlation sets  ccxxxii
CosNaming  xci, clxvi
create work area  ccxlxi
createFastPool() method  cclxxiv
credential vault  125
CSlv2  civ, cix–cx, clxxxii, clxxiv–cx, ccix
CSS Designer  cccxxxiii
currencies  52
custom object pool  cclxxiv
custom user registry  clxxvi
Customer Relationship Management (CRM)  cccxvii, 157
customers  52
CVS  cxxiv, ccv, cccxiv, cccxxviii, cccliii

D
DADX  cccxlvi–cccxlxi
   WebSphere Studio support  cccl
Data Definition view  cccxxix
Data description language (DDL) scripts  cccxxxi
data mapping  cclviii
data replication service  clxviii
data source  cxi, cxii, cxci–cxci, cciv
   V4 vs. V5  cxi
Database Web Pages wizard  cccxxxiv
DataDirect Technologies JDBC Drivers for Web-
   Sphere Application Server  xxx
DataSource object  cxc
DB Output view  cccxxxix
DB Servers view  cccxxxviii
DB2  xxx
DB2 XML Extender  cccxxxviii, cccxlxi–ccccl
dead beats  cclxvi
Debugger  ccclxii, cccliv
default messaging provider  ccxiv–ccxv
default URL provider  ccxiv
default_host virtual host  xcv, clxix
DefaultNodeGroup  clv
defered response  cclviii
development
   object pools  cclxxiii
discounts  52
discoverable services  110
Discovery Server portlet  151, 153
Discovery Services  156
discretionary response time  cxxxiii
dispatch rate  cxxxiv
distxcep.jar  ccclxxi
distributed server configuration  clxv
Document Access Definition eXtension
   see DADX
document management  115
Document Type Definition (DTD)  cccxxxii
Document Type Definition (DTD) Editor  cccxxxvii
documentum  112
domain expert  ccxiv
Domino  157
Domino Data Service  155
Domino server  119, 150
Domino Server Domain Index  120
Domino Web Access (iNotes) portlet  150, 152
Domino.doc portlet  120, 151–152
Dreamweaver  113
DTD  
   WebSphere Studio support  cccxxxvii
   DTD Generator  22
dynamic application partitioning  cxxi
dynamic cache  lxxviii, lxix, clxvi, clxix
dynamic cache replication  clxviii
dynamic caching service  cxxiv
dynamic cluster  cxxi, cxxxvi–cxxxvii, cxi
dynamic cluster manager  cxxvi
dynamic operation  cxxix–cxx
dynamic query  xlxi, cxxiii–cxliv, cxxvi
Dynamic Query Bean  cxxiv
dynamic workload manager  cxxvi
dynamically listen for JMS queues  cclxii

E
Eclipse  cccxvi–cccxx, cccxiv–cccxx
Edge components  xxix
Edge Side Include caching  xc, clxiv
EJB
high availability clxxii, cxcviii
high availability domain cxcix
high-end caching cxliii
high-performance computing cxxix, cxlii
HLASM ccclxxii, ccclxxv
HOD ccclxx, ccclxxii
HomePage Builder 113
Host-on-Demand xli, cccxxvii
hot standby cxcix
transaction log cxcix
HTML Syntax Validator cccxxxi
HTTP session persistence clxviii
Human Interaction support ccclxx

I
IBM Agent Controller ccclxiii
IBM Business Solutions xxx
IBM Content Manager 115, 120
IBM DB2 120, 157
IBM Directory Server 99–100, 122
IBM HTTP Server xxx, clviii
IBM WebFacing Tool xliii
ID Resolver 22
IIOP xlv
IMAP cxiii–cxiv, cxciii
IMS xli–xlii, xcix, clxxiv, ccxxxvi–ccxxxvii, ccclxx, ccclxxii, ccclxxiv
Information Integrator 121
Informix 157
initial context xci
default xci
instant messaging 101–102
internationalization (I18N) service lii
internationalization service
Web Services ccclxxvii
internationalized ccclxxvi
Interoperable Naming Service (INS) xcii, clxvii
Interwoven 112
inventory 52
Inventory subsystem 13
Invoke activity ccxxiii
iPlanet 122

J
J2EE Connector Architecture (JCA) lv, lxxviii, cxiv, clxii
J2EE Management ccix
J2EE perspective ccclxvii
J2EE publishing server ccclxv
J2EE Request Profiler ccclxi
J2EE security civ, clxxxv
JAAS lxxvii, civ, cvii, clxi, clxxxiii, clxxvii, ccix, 127
JACC clxxiii, clxxvii, ccix
JAF ccxxiv
Java 2 Platform, Enterprise Edition (J2EE) xxiv
Java 2 security lxxvii, ciii–civ, cvii, clxiii, clxiii–clxxiv, clxxvii
Java 2 Standard Edition (J2SE) xxiv
Java Authentication and Authorization Services (JAAS) cvii, clxxiv
Java Bean Web Pages wizard cccxxxiv
Java Browsing perspective ccxili
Java builder cccxlii
Java Community Process (JCP) 106
java comp/env cxvii
Java Connector Architecture (JCA) xl iv, xcix
Java Contract for Containers (JACC) clxxiv
Java debugger ccxis, ccclxxi
Java development tools cccxi, ccclxxii
Java development tools debugger ccxiv
Java Editor ccxii
Java Management Extension (JMX) ccx
Java Message Service ccxli
Java Message Service (JMS) cccxi
Java Profiling Agent cccxii
Java search ccxliii
Java Secure Socket Extension (JSEE) civ, clxxix
Java snippet activity ccxiv
Java tools for iSeries cccxxix
Java Type Hierarchy perspective ccxli
JavaBeans Activation Framework (JAF) cvxii, cccxii
Javadoc ccxiv
JavaMail cx, cxxiii–cxlxx, ccxii–ccxiii, ccxii
Built-in Mail Provider cxxiii, cxxii
JavaServer Faces cccxxv
JAXP ccx
JAXR ccxxii, ccxx
JAX-RPC ccxi, cccxxi, clxxxv–clxxvi, clxxx, ccxx
JCA ccx, cccxxi, cclxxix, cclxxx
CCI implementation ccxii, cccxii
Connection Manager ccxii–ccxiii
connection manager ccxii
resource adapter ccxii, cccxii
services lxxvii–lxxviii, clxiii–clxxii
Web services ccxix
JCA connector clxxiii
JCA Web services ccxi, clxxiv
JCL ccclxxiii, ccclxxv  
JDBC ccix  
JDBC driver cxci  
JDBC provider cx, cxci  
JDBC resource provider cxi, cxci  
JFace ccxxviii  
JMS ccix, cclxxi, cccxxi–ccclxxii  
JMS client clxxxi  
JMS client classes (MA88) cxv  
JMS destination xc, clxiv  
JMS provider cxi, cxv, cxci, cxxiv  
JMS pull model cxlvii  
JMS server lxxxi–lxxxii, xciv, cxv–cxvi, cxx–cxxi, clivii  
architecture lxxxi  
definition lxxx  
JMX lxxvii, xciii, cv, cxx, cxii, clxvii, cxxiv, ccxv, ccli  
JMX MBeans cx, cxc  
JNDI xci, cxvii, cxxii, cxxiv, clxiv–clxvi, cxvii, ccxiv, ccli  
provider URL xcii, clxvii  
Job Monitor Server ccclxxvi  
JSESSIONID xcb, clxx  
JSR 101 xci, clxxiii  
JSR 109 xci, clxxiii, clxv, ccxi  
JSR 168 107  
JSR-047 ccclvi  
JSR-077 xci, clxvii  
JSR-109 clxx  
JTA ccix  
JUnit ccclxxii  
jurisdictions 52

L
languages 52  
launchClient command lxxvii  
LDAP civ–cv, clxxxiii, clxxxvi, ccix, ccclxxi, 9, 12,  
120, 122, 124  
LDAP server xxx  
Light Weight Third Party Authentication (LTPA) cvii, clxvii  
Lightweight Third Party Authentication (LTPA) 124–125  
listener port clxxii  
Load At policy ccli  
Load Balancer cxi, ccx  
loader 23  
Loader Package 22–23  
Loader Packager 22  
Log Analyzer cxx, ccclvi  
Log and Trace Analyzer ccclvi  
Logging tool ccclvi  
Lotus Collaborative Components 155  
Lotus Discovery Server 153–155  
Lotus Domino 122, 155  
Lotus Extended Search 100, 119  
Lotus Instant Messaging (Sametime Chat) portlet 152  
Lotus Instant Messaging and Web Conferencing  
(Sametime) 153–154  
Lotus Instant Messaging portlet 151  
Lotus Notes 120  
Lotus Notes and Domino portlet 152  
Lotus QuickPlace 155  
Lotus Sametime 155  
Lotus Team Workplaces (QuickPlace) 153  
Lotus Team Workplaces (QuickPlace) portlet 151–152  
Lotus Workplace Content Development 112  
LPEX Editor ccclxxxii  
LTPA cvii, clxxxiii, clxxxvii, ccix, 126–127

M
mail.jar cxiv, cxciii  
managed data 52  
managed node clvii  
managed server cxx  
manual mode cxlii  
Map Designer lxii  
Marketing subsystem 13  
master configuration repository cxxi  
mediation clxvii  
member security services 10  
member services 121  
Member subsystem 8, 10, 12  
user registration 8  
members 52  
memory-to-memory session persistence xcb, clxvii  
Merchandizing subsystem 13  
Message aggregation ccclxxix  
Message Brokers Toolkit lvii, ccclxxviii, cxcxi, ccx- 
civ, ccxcvi–ccxcvii, ccccc–ccx, cccxii–ccxvii, ccxxii–ccxxix  
message enrichment ccclxxvii  
message flow applications ccxcvi  
message flow debugger ccxvii
message flow deployment  ccxx
message flow development  ccxci, ccxciv, ccxcvii, ccxiv
message flow nodes  ccxcvi
message flows  ccxc, ccxcv
message formats  cclvii
message listener  lxxvii, xc, clxi, clxiv
message node types  ccci
message routing  cclxxix, ccxcvi
Message Sending pattern  cclix
message set  ccxc, ccxcv
message store  clxxx
message transformation  cclxxxvii–cclxxxix, ccxcvi
message-driven bean  clxxxi, ccxii, ccix
message-driven beans  lxxxii, cxxv, cclvii
messaging bus  clxviii
messaging engine  clxix, clxxxi
messaging patterns  cclvii
Messaging Receiving pattern  cclix
Messaging subsystem  13
Metadata  ccxlxi
Microsoft Active Directory  122
Microsoft Index Server  120
Microsoft SharePoint  120
Microsoft Site Server  120
Microsoft SQL Server  120, 157
MIME  cxiv, cxciii
Model-View-Controller (MVC)  cccxxv
MQ  9
MQ Multicast Transport  cccvii
MQ Real-Time Transport  cccvii
MQSeries Java Client SupportPac, MA88  cccvii
multicast JMS clients  ccxci
My Team Workplace portlet  153

N
name server  xci, clxvi
name service
  name bindings  xci
name service (JNDI)  lxxxvii, clxi
name space  xcii, clxvi
naming service
  name bindings  clxvi
NEONMSG  cclxxxix
nested work areas  cclxxxii
Netegrity SiteMinder  124, 127
Netscape Directory Server  122
New Era of Networks Rules and Formatters  cclxxx-
ix
node
  clustering  lxxxii, clvii
    definition  lxx, clv
node agent  xci–xcii, xciv, cxix–cxx, cxxxvi, clxv–clx-
v, cci
    definition  cxxi, cci
node group  cxxi, cxxvii, cxi, clv
Non-classifier rule  cclvii
non-JMS middleware  cclxii
Notes and Domino portlet  150
Novell eDirectory  122

O
object pool manager  cclxxiii
object pools  lxi, cclxiv–cclxxv
  development  cclxxiii
object query statement  ccxlxi
Object Request Broker (ORB)  xc, clxiv
Object Request broker (ORB)  lxxxx, xc, clxi, clxiv
object usage frequency  cclxxiii
objectpool.jar  cclxxiii
on demand mode  cxlii
on demand operating environment  cxviii
on demand router  cxxxi–cxxxiv, cxl
On demand router for HTTP  cxvii
online registration  9
online transaction processing (OLTP)  cxliii
Open Content Syndication (OCS) channels  112
Operating system security  ciii
operating system security  cclxxiv
operational data  52
operational policy  cxxxi
Oracle  120, 157
ORB  cxc
order subsystem  13
Organization for the Advancement of Structured In-
formation Standards (OASIS)  106
OS/390 Foreign File System server  cccxxv
Output view  ccxl

P
Package Explorer view  ccxl
Page Designer  cccxxxi
parallel task  clxii
partition  clxlvi, clxviii
partition tasks  cllix
partition-aware workload management  clxiv, clxvi
partitioning cxlv
partitioning pattern cxlili–cxlv
partner link ccxviii, ccxvi
pass data between application components cclxxix
payment 52
peer-to-peer topology clxviii
People Finder portlet 151, 153
People Service 155–156
PeopleSoft 157
performance goals cxxxiii
Performance Monitoring Infrastructure (PMI) lxxvii, xciii, clxvii
Performance Monitoring Instrumentation (PMI) ccx
Performance Monitoring Interface (PMI) clxi
Persistence Manager ccxxxvii
persistent area clxvi
Personalization 99
Perspective cccxix, cccxxi
Pervasive computing cccxxxiii
pervasive devices ccxxi
PhotoShop 113
Pick activity cxxxi, cccxx
PL/I cccxxxiv–ccclxxv
PL/I tools ccclxxv
pluggable authentication module cvi, clxxxvi
pluggable authorization interface cvii
plug-in ccxxvii
Plug-in Development Environment (PDE) cccxxix
Plug-in development environment (PDE) cccxxviii
PMI request metrics xciii, clxviii
PMI service xciii, clxviii
point-to-point ccxxvi
PoolableObject ccclxxiv
pooled connection ccxci
POP3 cxiii–cxiv, cccxxiii
port
  admin_host virtual host xcv, clxx
  default host xcv, clxix
port type ccxvii, cccxi
Portal
  Administration 143
  Analysis and logging 149
  Cascading 132
  Mobile 160
  Personalization 130
  Settings 144
  Virtual 137
Portal Document Manager 99–100
Portal Document Manager (PDM) 115
Portal Server 99–100
Portlet
  API 105
  Applications 104
  Catalog 147
  Collaborative 150
  cooperation 107
  installing 146
  managing 145
  modes 105
  performance 106
Portlets 102
prepare commit ccxlv
prepareQuery() method ccxlv
prices 52
Principals clxxxvii
Private UDDI Registry lxxvii, ccxi
Process Choreographer I
Process Debugger ccxix, ccclxxi
Process Designer lxi
Process Editor ccxvi, ccclxx
process instance cccxxxix
process template cccxxix
Productivity Components 100, 116
profiling priorization cccxxix
Profiling tool ccclvi, ccclxx
Program editor ccclxxv
Programming Model Extensions clxi
programming model extensions xxviii
protocol converter ccxvii
protocol provider cxiii, ccxiii
publish/subscribe ccclusxx–ccclxxxix, ccxci, ccxxiv,
ccxxvii–ccxviii, cccxxvi, cccxxvii, cccxxvii, cccxxvii, cccxx
push down technology ccclxxii
Q
quality of service cxxix, clxxxii
Query by Image Content (QBIC) 121
Query Engine ccxlv
Query enhancements cccxxii
Query support ccxlii
QueryBean ccxlv
queue length cccxiv
QuickPlace 100, 102
QuickPlace Service 156
R
RACF cccvii
rapid application development (RAD) cccxii
Rapid Deployment Wizard lxiv
Rational Application Developer xxx, clxxv, ccv–ccvi, ccviii
Rational ClearCase cxxiv, ccv
Rational ClearCase LT cccxxiii, cccxxviii, cccliii
Rational Developer xcvii
Rational Rose cxxiii–cxxiv, ccv
Rational Web Developer xxx, ccviii
Rational XDE cxxiii, ccv
RDB to XML Mapping Editor cccxxxviii
real-time JMS clients cxxcii
Receive activity cxxciii, cxxxx
Receive bean cclviii
Redbooks Web site 167
Contact us xix
refactoring cccx
refactoring support cccxlii
referenceable ccxvi
relational database
WebSphere Studio support cccxxxviii
relational database to XML mapping cccxxxviii
Relational Resource Adapter cccxxvii
relational resource adapter cxii–cxiii, cxcii
Relationship Designer lxii
Remote System Explorer (RSE) xliii
Remote System Explorer (RSE) perspective cclxxx
Reply activity cxxciii, cxxxx
report generators cccxiii
resource adapter cxi, cxxiv, cxci, ccvi
resource environment entry cxxvi
resource environment provider cxi, cxxvii, cxci, ccvii
resource providers cx, cx
resource provisioning cxxciii
resource sharing cxxvii
response-time goals cxxciii
returnObject() method cclxxiv
REXX xlii, cccxxix
REXX/370 Alternate Library cccxxvi
Rich Site Summary 112
RMI/IIOP xlv, lxxxiv, ccxxviii
Rollout Update clvii
round robin routing policy lxxxiv, ccviii
RRA cccxxvii
Rule Implementation Developer cclv
Rule loader cclv
Rule Management cclv
Rule Management Application cclv
Rule Management tool cclv
Rule object cclvi
Rule-based Application Component Developer cclv
RuleImplementor cclvi
Rules & Formatter Extension cclxxxix
run-time map cxl

S
SAAJ ccix
Sametime 100–102
Sametime Contact List portlet 153
Sametime portlet 151
Sametime Service 156
Sametime Who Is Here portlet 153
SAP 157
SAS interceptor cx, cxc
SCADA (Supervisory, Control And Data Acquisition) cccvii
SCADA clients cccxii
scalability clxxxii
Scheduler service
Wakeup daemon cclxix
scheduler service liii
alarm thread cclxix
runtime cclxix
SCLM cccxxiv
SCM cccxxiii–cccxxiv
Scrapbook cccxxiv
Scrapbook Page Editor cccxxiv
Security
WebSphere Portal 121
security
Web services xci, clxxiii
WebSphere Application Server lxxvii, xciv, ciii, cvi–cix, clxi, clxix, clxxiii, clxxvii–clxxix, ccix
WebSphere Business Integration Message Brok er cxxcii, cccvii
WebSphere Commerce 10, 16
WebSphere UDDI Registry ci, clxxvi
security collaborator cclxxix
security server cclxxviii
sellers 52
Sequence activity cccxviii
sequence-based process cxxc, cccxii
Server Access Interface lv
Server tools ccvi, cccxxiii
server weighted routing policy lxxxiv, ccviii
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server-side JavaScript debug adapter</td>
<td>ccclv</td>
</tr>
<tr>
<td>Service Data Object (SDO)</td>
<td>ccix</td>
</tr>
<tr>
<td>service integration bus</td>
<td>clxxviii-clxxxv, clxxx-clxxxii</td>
</tr>
<tr>
<td>service policy</td>
<td>cxxvii, cxxxv, cxl-cxli</td>
</tr>
<tr>
<td>Service-Oriented Architecture (SOA)</td>
<td>xlvii</td>
</tr>
<tr>
<td>Service-oriented development</td>
<td>cccxxvi</td>
</tr>
<tr>
<td>Services-oriented architecture</td>
<td>ccclx</td>
</tr>
<tr>
<td>session control</td>
<td>11</td>
</tr>
<tr>
<td>session management</td>
<td>xcv, clx, clxx</td>
</tr>
<tr>
<td>JSESSIONID</td>
<td>xcv, clxx</td>
</tr>
<tr>
<td>SSL session identifiers</td>
<td>xcv, clxx</td>
</tr>
<tr>
<td>session persistence</td>
<td>xcvi, clxxi</td>
</tr>
<tr>
<td>session types</td>
<td>12</td>
</tr>
<tr>
<td>set of alarms</td>
<td>cclxvi</td>
</tr>
<tr>
<td>setProperties() method</td>
<td>cclxxiv</td>
</tr>
<tr>
<td>Shared Work Area</td>
<td>iii</td>
</tr>
<tr>
<td>Shared Work Area service</td>
<td>cclxxix-cclxxxi</td>
</tr>
<tr>
<td>shipping</td>
<td>52</td>
</tr>
<tr>
<td>Siebel</td>
<td>157</td>
</tr>
<tr>
<td>Simple Object Access Protocol (SOAP)</td>
<td>lvii</td>
</tr>
<tr>
<td>Simple WebSphere Authentication Mechanism</td>
<td>swam</td>
</tr>
<tr>
<td>(SWAM)</td>
<td>cvi, clxxvi, ccix</td>
</tr>
<tr>
<td>Single sign-on</td>
<td>125</td>
</tr>
<tr>
<td>single sign-on</td>
<td>cvii, clxxxvii, 12</td>
</tr>
<tr>
<td>singleton services</td>
<td>cclxvii</td>
</tr>
<tr>
<td>Site Map editor</td>
<td>cccxxx</td>
</tr>
<tr>
<td>Site Parts palette</td>
<td>cccxxxv</td>
</tr>
<tr>
<td>SMTP</td>
<td>cxiii-cxiv, cxiciii</td>
</tr>
<tr>
<td>SNMP</td>
<td>cviii, ccxix</td>
</tr>
<tr>
<td>SOAP</td>
<td>ccxi</td>
</tr>
<tr>
<td>SOAP over HTTP</td>
<td>cii</td>
</tr>
<tr>
<td>SOAP with Attachments API for Java (SAAJ)</td>
<td>clxxiii</td>
</tr>
<tr>
<td>soapaarenable</td>
<td>c</td>
</tr>
<tr>
<td>Software configuration and library manager (SCLM)</td>
<td>cccliv</td>
</tr>
<tr>
<td>Software Configuration Library Manager (SCLM)</td>
<td>cccxiii</td>
</tr>
<tr>
<td>Software Configuration Management (SCM)</td>
<td>cccxx, cccxiii</td>
</tr>
<tr>
<td>specific finders</td>
<td>ccxliii</td>
</tr>
<tr>
<td>SQL statement</td>
<td>ccxliii</td>
</tr>
<tr>
<td>SQL stored procedure debugger</td>
<td>ccclv</td>
</tr>
<tr>
<td>SQLJ debugger</td>
<td>ccclv</td>
</tr>
<tr>
<td>SQLJ File wizard</td>
<td>cccxl</td>
</tr>
<tr>
<td>SQLToXML</td>
<td>cccxxxviii</td>
</tr>
<tr>
<td>SSL</td>
<td>ci, clxxvi, 126</td>
</tr>
<tr>
<td>SSL session identifiers</td>
<td>xcvi, clxx</td>
</tr>
<tr>
<td>Staff activity</td>
<td>ccxiii, cccxx</td>
</tr>
<tr>
<td>Stale beats</td>
<td>cclxvi</td>
</tr>
<tr>
<td>Standard Java security</td>
<td>ciii, clxxxiv</td>
</tr>
<tr>
<td>Standard Widget Toolkit (SWT)</td>
<td>cccxviii</td>
</tr>
<tr>
<td>standards</td>
<td></td>
</tr>
<tr>
<td>ANSI X12</td>
<td>lxvi</td>
</tr>
<tr>
<td>EDIFACT</td>
<td>lxvi</td>
</tr>
<tr>
<td>HIPAA</td>
<td>lxvi</td>
</tr>
<tr>
<td>HL7</td>
<td>lxvi</td>
</tr>
<tr>
<td>RAIL</td>
<td>lxvi</td>
</tr>
<tr>
<td>UCS</td>
<td>lxvi</td>
</tr>
<tr>
<td>VICS</td>
<td>lxvi</td>
</tr>
<tr>
<td>Startup beans</td>
<td>li</td>
</tr>
<tr>
<td>startup beans</td>
<td>cclxx-cclxxi</td>
</tr>
<tr>
<td>startup service runtime flow</td>
<td>cclxxi</td>
</tr>
<tr>
<td>stateful session EJB persistence</td>
<td>clxviii, clxxii</td>
</tr>
<tr>
<td>Static Web publishing server</td>
<td>cccxlv</td>
</tr>
<tr>
<td>Stellent</td>
<td>112</td>
</tr>
<tr>
<td>Store</td>
<td>39-41, 49-52, 58, 72-73</td>
</tr>
<tr>
<td>Store front assets</td>
<td>40</td>
</tr>
<tr>
<td>Stream</td>
<td>cxxii</td>
</tr>
<tr>
<td>stream</td>
<td>cciv</td>
</tr>
<tr>
<td>Structured Query Language (SQL)</td>
<td>cccxiv</td>
</tr>
<tr>
<td>Struts</td>
<td>xlii, cccxxv, cccxxxv-ccccxxxi, cccxxxi, 107</td>
</tr>
<tr>
<td>Struts Configuration File Editor</td>
<td>cccxxvi</td>
</tr>
<tr>
<td>Struts tools</td>
<td>cccxviii</td>
</tr>
<tr>
<td>Subject class</td>
<td>clxxvii</td>
</tr>
<tr>
<td>SubsystemMonitor</td>
<td>cclxiv, cclxvi</td>
</tr>
<tr>
<td>SubSystemMonitorManager</td>
<td>cclxv</td>
</tr>
<tr>
<td>SubSystemMonitorManager</td>
<td>cclxv</td>
</tr>
<tr>
<td>subsystems</td>
<td>8</td>
</tr>
<tr>
<td>supervisory mode</td>
<td>cxlii</td>
</tr>
<tr>
<td>Supply Chain Management</td>
<td>cccxxvii</td>
</tr>
<tr>
<td>supported units of measure</td>
<td>52</td>
</tr>
<tr>
<td>SWAM</td>
<td>cvi, clxxiii, clxxxvii</td>
</tr>
<tr>
<td>Swing</td>
<td>ccclii</td>
</tr>
<tr>
<td>Switch activity</td>
<td>ccxiv</td>
</tr>
<tr>
<td>Sybase</td>
<td>120</td>
</tr>
<tr>
<td>Symptom database</td>
<td>cclix</td>
</tr>
<tr>
<td>synchronization</td>
<td>cclix</td>
</tr>
<tr>
<td>synchronous response</td>
<td>cclviii</td>
</tr>
<tr>
<td>Syndicated content</td>
<td>111</td>
</tr>
</tbody>
</table>

**T**

<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>task</td>
<td></td>
</tr>
<tr>
<td>partition</td>
<td>cclxii</td>
</tr>
<tr>
<td>taxes</td>
<td>52</td>
</tr>
<tr>
<td>TCP/IP monitoring server</td>
<td>cccxlv</td>
</tr>
<tr>
<td>team collaboration</td>
<td>cccliii</td>
</tr>
<tr>
<td>team development</td>
<td>cccxiv, cccxxiii</td>
</tr>
<tr>
<td>Team Workspaces</td>
<td>151</td>
</tr>
</tbody>
</table>

178  WebSphere Product Family Overview and Architecture
telemetry messages cccvii
template cccxxxv
Terminate activity cccxiv
terminate work area cclxxxii
Text Transform tool 22
Third-party authentication servers 124
Throw activity cccxiv
Tivoli Access Manager clxxxv, 124–125, 127
Tivoli Access Manager Servers for WebSphere Application Server xxxi
Tivoli Access Manager WebSeal 124
Tivoli Directory Server for WebSphere Application Server xxx
Tivoli Intelligent Orchestrator cxxvii
Tivoli Performance Viewer xciv, clxviii, cxx
Tivoli Provisioning Manager cxxxiv, cxxvii
Tivoli Web Site Analyzer 100, 142, 149
Tivoli Intelligent Orchestrator cxxxiv
Tomcat xl
topic cccxi, cccxvii, cccxiv–cccvii, cccx
topic-based filtering cccvi
topic-based message routing cclxxxviii–cclxxix
topic-based security cccxi
Trading subsystem 13
transaction cccxlvi
Transaction class cxxxii
transaction class cxxxii–cxxxiii, cccx, cxx
transaction management lxxxii, lxxxvii, clxi
transaction service lxxviii, clxii
Transformer activity cccxiii
transient area clxvi
transport chain clx
treemap clx–clxi
trigger point ccliii
Trust Association Interceptor (TAI) 124
TSO xlii, ccclxiii
TSO Command Server ccclxvi
Type safe notification cclxxv

U
UDDI lixiv, lxiv, xcviii, ciii, clxxii, clxxvii, cccxlix–cccl, 148
UDDI Registry clxxvi
UDDI registry c
UML cccxlvi
UML visual editing cccxlviii
Unified Change Management (UCM) cclxiv
Unified Modeling Language (UML) cxxiii, ccv
unit of work scope cclxvii
Universal Description Discovery and Integration (UDDI) ccclix
Universal Test Client cclxvi–cclxvii
unmanaged Web server node clvii
URL provider cxi, cxiv, cxxi, cxciv
URL registry 51
URL rewrting xcv–xcvi, clxx, 12
Use work area cclxxxi
User Name Server cxxi, cccxiv, cccviii
user registration methods 8–10
user registry cv, clxxxv
UserWorkArea cclxxxi

V
V5 default messaging provider cccxiv
Value object methods cccxliii
variable cccxv
view cccxi
view registry 52
Vignette 112
Virtual host
   admin_host xcv
   default_host xcv
virtual host cxxv, ccxvii
   admin_host clxii
   architecture xcv, clxii
   default_host xcv, clxii
virtualization cxx
Visual Editor for Java cccxliv–ccclxvi
visualization cccxxviii

W
Wait activity cccxiv
Wake up daemon thread cclxix
wasStartupPriority cclxxi
wasStartupPriority property cclxxi
Web clipping portlet 144
Web Conference center 151
Web Conferencing (Sametime) portlet 151
Web Conferencing portlet 153
Web container lxxviii–lxxxi, cix, cxxii, clxv, cclxxix, cccxiv–cccvii, ccxvi–cccvii, ccv, cccxlxi
   See also J2EE Web container
   override lixxxvii, clxvii
Web container inbound chain clx
Web Deployment Descriptor editor cccclxxvii
Web Diagram Editor cccclxxvi
subsystems 8
   Catalog subsystem 12
   Inventory subsystem 13
   Marketing subsystem 13
   Member subsystem 8
   Merchandising subsystem 13
   Messaging subsystem 13
   Trading subsystem 13
WebSphere Commerce Payments Server 2
WebSphere Commerce Portal 160
WebSphere Commerce Server 2
components
   subsystems 8
WebSphere Commerce Server instance 41
WebSphere Data InterChange lxvi
WebSphere Data Interchange lxvi, 167
WebSphere dynamic caching service cxlv
WebSphere Enhanced EAR ccvii
WebSphere Event Broker cxvi
WebSphere Extended Deployment cxxvii
WebSphere extensions cccxxviii
WebSphere InterChange Server lii–lv, lvii–lx
   overview lii
   platform support lii
WebSphere internal messaging xcvi
WebSphere JMS provider lxiii, cv–cxvi
WebSphere MQ xxviii, lvii, cv–cxvi, cxxvii, clxxi,
   ccxx–cxv, ccxci, cccv, cccvii
WebSphere MQ Application Messaging Interface (AMI) cccvi
WebSphere MQ clients ccxci
WebSphere MQ cluster ccxiv
WebSphere MQ Enterprise Transport cccxi, cccvii
WebSphere MQ Event Broker cclxxvii
WebSphere MQ Everyplace cccvii
WebSphere MQ Everyplace clients ccxci
WebSphere MQ Integrator cclxxvi
WebSphere MQ Integrator Broker for Multiplatforms lii
WebSphere MQ JMS provider cv–cxvi, ccxvi–ccxv
WebSphere MQ MA88 SupportPac cxvi
WebSphere MQ Message Queue Interface (MQI) cccvi
WebSphere MQ Mobile Transport cccxi, cccv, cccvii
WebSphere MQ Multicast Transport cccvii
WebSphere MQ Object Authority Manager cccvii
WebSphere MQ Real-time Transport cccxi, cccvii
WebSphere MQ Telemetry Transport cccxi, cccvii
WebSphere MQ Web Services Transport cccxi, cccvii
WebSphere MQ Workflow lii, lxii–lxiii
   overview lxii
WebSphere MQ Workflow Buildtime lxiii
WebSphere MQ Workflow Web-services Process
WebSphere MQ Workflow Buildtime lxiii
WebSphere partition facility cxxvii–cxxviii, cxlv–clxxix
WebSphere Personalization 140–141
WebSphere Portal xxiii, lxx–lxii, lxxiii
   architecture 97–98
   product editions 98
WebSphere Portal Application Integrator (WPAI) 157
WebSphere Portal Content Publisher 157
WebSphere Portal Content Publishing (WPCP) 113
WebSphere Portal Enable 99
WebSphere Portal Express 100
WebSphere Portal Express Plus 101
WebSphere Portal Extend 99
WebSphere Portal for Multiplatforms lxx
Websphere Portal Search Engine Indexes 120
WebSphere Process Choreographer 114
WebSphere Rapid Deployment cccvi–cccv
WebSphere Relational Resource Adapter ccxvi
WebSphere Studio xxiii, xxxvii, xcix–c, clxxiv
   functional comparison cccxxviii
   overview cccxi
   Platform support xxxix
WebSphere Studio Application Developer cccxxvi
   Overview xli
   team collaboration cccxxvii
WebSphere Studio Application Developer Integration
   Edition xlvi, cccxxix
   description cccxxvi
   overview xli
WebSphere Studio Asset Analyzer (WSAA) cccxxvii
WebSphere Studio Client for iSeries tools cccxxvii
WebSphere Studio Enterprise Developer cccxxix
   description cccxxvii
   overview xlii
   team collaboration cccxxix–ccclvii
WebSphere Studio Site Developer 99–100, 110
   description cccxxiv
   overview xli
WebSphere Product Family Overview and Architecture

Discover the WebSphere family

This collection of papers is intended to help you understand the products that make up the WebSphere family. The first paper gives you a big picture overview of the WebSphere product family and how the products interrelate. Following are papers that discuss individual product or product families in more detail.

Take an in-depth look at key products

This collection will be useful to IBMers, customers, and business partners that are researching solutions to business problems. It is aimed at decision makers and architects.

Compare capabilities

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information: ibm.com/redbooks