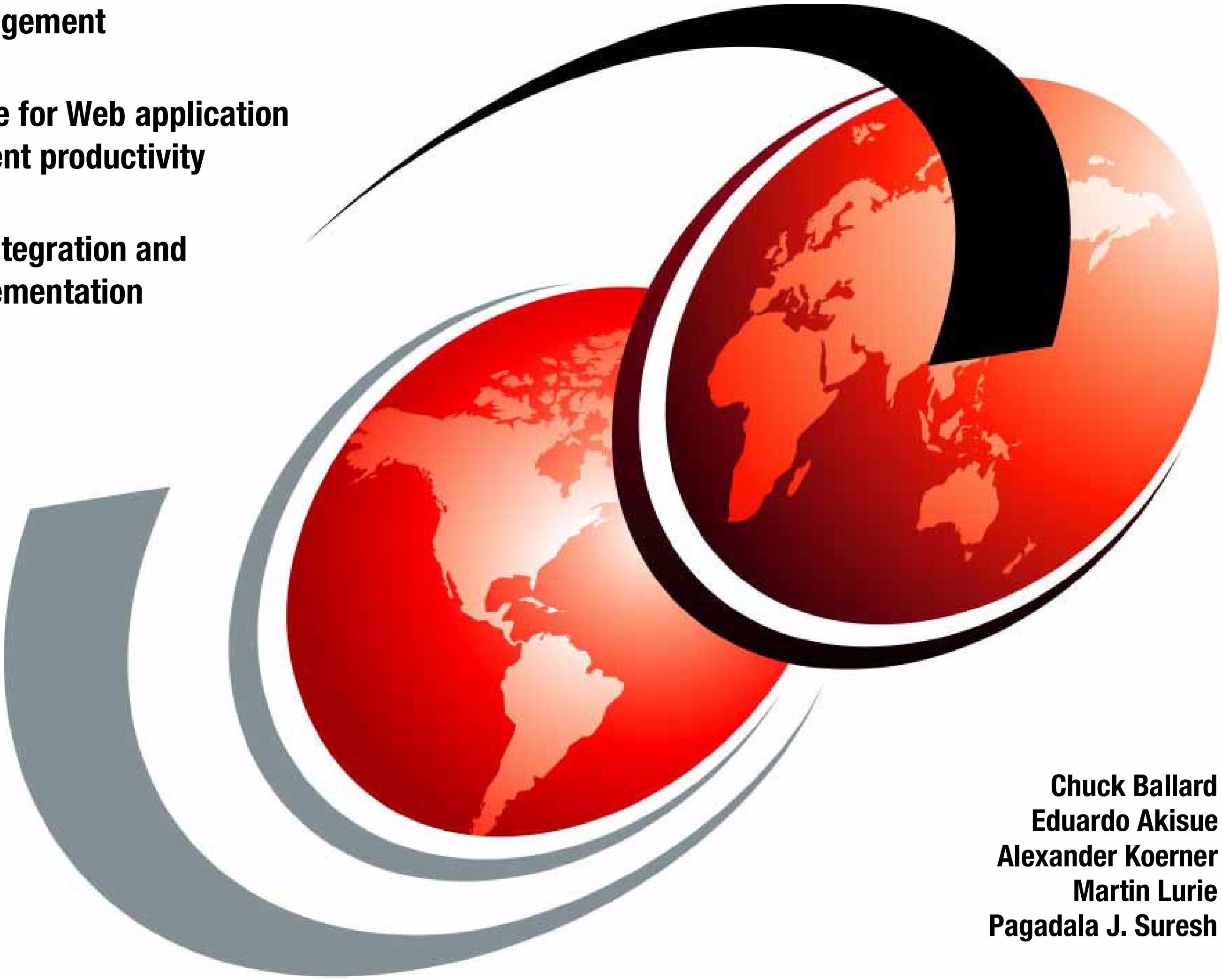
Using Informix Dynamic Server with WebSphere

Informix Dynamic Server for data management

WebSphere for Web application development productivity

Solution integration and easy implementation



Chuck Ballard
Eduardo Akisue
Alexander Koerner
Martin Lurie
Pagadala J. Suresh

Redbooks

ibm.com/redbooks

International Technical Support Organization

Using Informix Dynamic Server with WebSphere

June 2003

SG24-6948-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (June 2003)

This edition applies to Informix Dynamic Server Version 9.4, WebSphere Application Server Version 5, and WebSphere Studio Application Developer Version 5. Windows/2000 was used as the operating environment for WebSphere Studio, and SuSE Linux V8.0 was used as the operating environment for IDS and WebSphere Application Server.

Contents

Notices ix
Trademarks x

Preface xi
Executive summary xiii
The sample applications. xiv
Highlights and benefits xiv
The team that wrote this redbook. xv
Become a published author xviii
Comments welcome. xviii

Introduction. xix

Chapter 1. Informix Dynamic Server: An overview 1
1.1 Informix database family 2
1.2 Understanding Informix Dynamic Server (IDS) 3
1.3 Feature highlights 8

Chapter 2. Installing and configuring IDS V9.40 21
2.1 Taking advantage of new IDS 9.40 features 22
2.2 Installing IDS 9.40 on SuSE Linux V8 31
2.3 Configuring IDS for use with WebSphere 56

Chapter 3. WebSphere V5: An overview 63
3.1 What is WebSphere? 64
3.2 WebSphere product family 67
3.3 WebSphere Application Server (WAS) 68
 3.3.1 J2EE: Overview. 68
 3.3.2 WebSphere Application Server: Architecture 71
 3.3.3 WebSphere Application Server: Packages 73
3.4 WebSphere Studio Application Developer (WSAD) 77
 3.4.1 Role-based development model 78
 3.4.2 WebSphere Studio Application Developer - Packages 80
3.5 WebSphere Studio integration with Informix 4GL 84

Chapter 4. Installing and configuring WebSphere Studio V5 85
4.1 Installing WSAD on Windows/2000. 86
4.2 Configuring WSAD for IDS 87
4.3 A sample application using Database Web Pages 88

4.3.1 Deploying the application in WebSphere Application Server 102
4.3.2 Summary 102

Chapter 5. Installing and configuring WebSphere Application Server . 103
5.1 Installing WAS on SuSE Linux V8.0 104
5.1.1 Verifying the installation 111
5.2 Configuring WAS for use with IDS 114
5.2.1 Configuring the Informix JDBC Provider 114
5.2.2 Verifying the configuration. 115
5.3 WebSphere Administration Repository 118

Chapter 6. Integrating IDS and WebSphere 121
6.1 Scenarios for deploying IDS and WebSphere. 122
6.1.1 Implementation scenarios 123
6.2 High availability considerations 125
6.2.1 IDS high availability and WebSphere 127
6.2.2 High availability with HDR 127
6.2.3 JDBC support for IDS and HDR 128
6.3 Systems architecture 129
6.4 Connecting IDS and WebSphere 130
6.4.1 Using a Data Source object 131
6.4.2 Configure Informix Data Source on WebSphere Studio 132
6.4.3 Configure Informix Data Source on WebSphere Application Server136

Chapter 7. Working with IDS and WebSphere 143
7.1 Introduction to the sample applications. 144
7.2 Extending the Database Web Pages sample application 144
7.2.1 Steps to extend the sample application 145
7.2.2 Creating a JSP to integrate the application modules 149
7.2.3 Deploying the application 151
7.3 Sample application: Container Managed Persistent Bean 152
7.3.1 The ITSOStores sample application 152
7.3.2 Steps for creating a CMP bean. 152
7.3.3 Deploying the application 154
7.4 A sample application using JMS and IDS 158
7.4.1 The ITSOStoresJMS sample application 159
7.4.2 Preparing the sample application for deployment. 160
7.4.3 Store and forward mechanism 163
7.5 Managing transactions 165
7.5.1 Java Transaction Service (JTS) 166
7.5.2 Local and global transactions in WebSphere 167
7.5.3 Bean-managed transactions (BMT) 168
7.5.4 Container-managed transactions (CMT). 168
7.5.5 Transactional programming considerations 171

7.5.6 General guidelines for using transactions 172

Chapter 8. IDS, WebSphere, and XML 175
8.1 An introduction to XML 176
8.1.1 XML usage scenarios 177
8.2 IBM Informix Dynamic Server (IDS) and XML. 179
8.2.1 XML support in database systems 180
8.2.2 What's available with IDS 181
8.3 Dynamic XML mapping with WSAD V5 and IDS. 186
8.3.1 XML tools in WSAD V5 186
8.3.2 The SQLtoXML and XMLtoSQL framework in WSAD 187
8.3.3 Create a wizard based SQLtoXML sample project 188
8.3.4 Enhance the sample project with the XMLtoSQL class library 200
8.3.5 Additional dynamic XML mapping options in WSAD V5 204

Chapter 9. IDS, Web services, and WebSphere 209
9.1 Introduction to Web services 210
9.1.1 Web service standards 211
9.1.2 WSAD V5 tools for Web services development 213
9.2 Using IDS as a Web service provider 214
9.2.1 IDS 7/9 Web services based on EJBs 214
9.2.2 IDS 7/9 Web services based on Java beans 215
9.2.3 DADX Web services and IDS 7/9 221
9.3 Using IDS 9 as a Web service consumer 233
9.3.1 IDS 9 and J/Foundation 235
9.3.2 Installation and configuration of IDS 9 and AXIS for the examples. 236
9.3.3 The basic IDS Web service consumer development steps 238
9.3.4 The AXIS WSDL2Java tool 239
9.3.5 A simple IDS Web service example — Currency Exchange project 239
9.3.6 A complex IDS Web service example — Google search 245

Chapter 10. WebSphere Portal Server 253
10.1 An introduction to WebSphere Portal 254
10.1.1 WebSphere Portal Experience 256
10.1.2 Industry impact and acceptance 257
10.2 WebSphere Portal architecture 258
10.2.1 WebSphere Portal tooling 261
10.3 WebSphere Portal 261
10.3.1 Portal concepts 261
10.3.2 Portlets 262
10.3.3 Portlet lifecycle 267
10.3.4 Portlet events and messaging. 268
10.3.5 Page aggregation 270
10.4 Portlet solution patterns. 273

10.5 IDS and WebSphere portal server 276
10.5.1 Configuring IDS for WebSphere Portal 277

Chapter 11. WebSphere MQ, messaging, and IDS 279
11.1 WebSphere MQ overview 280
11.2 WebSphere and messaging 283
11.2.1 Overview 283
11.2.2 Java Message Services (JMS) 283
11.2.3 WebSphere Messaging Engines. 283
11.3 WebSphere MQ Integration with IDS 285
11.3.1 MQSeries Transactional Support for IDS 286
11.3.2 IBM Informix MQSeries DataBlade 286

Chapter 12. IBM Informix 4GL and WebSphere 289
12.1 IBM Informix 4GL: Protecting your investment 290
12.1.1 Informix 4GL 290
12.1.2 EGL and WebSphere 291
12.1.3 Extending EGL to support I4GL 292
12.1.4 Moving from I4GL to EGL 294
12.1.5 The value of WebSphere Application Developer 294

Chapter 13. Implementation hints and tips. 295
13.1 Our implementation experience 296
13.1.1 Installing SuSE 8.0 Linux 296
13.1.2 Installing IDS on SuSE 8.0 Linux 298
13.1.3 Performance tuning guidelines 303
13.1.4 Determining the port number of IDS on Linux. 305
13.1.5 Using sequence objects rather than serial data type 307
13.1.6 WebSphere and IDS 307
13.1.7 Install error with Redhat 8.0 Linux. 308
13.1.8 An alternative Java UDR deployment method 308

Appendix A. SQLtoXML and XMLtoSQL Java class description 311
Class com.ibm.etools.sqltoxml.SQLToXML 311
Constructors 311
Methods 311
Class com.ibm.etools.xmltosql.XMLToSQL 317
Constructors 318
Methods 318
Class com.ibm.etools.sqltoxml.QueryProperties 323
Constructor overview 323
Method overview 323
Class com.ibm.etools.xmltosql.SQLProperties 324
Field overview 324

Constructor overview 325
Method overview 325
Class com.ibm.etools.sqltoxml.BaseProperties 325
Constructor overview 325
Method overview 326

Appendix B. DADX file format 327

Appendix C. IDS and WSAD on Windows 333
Configuring IDS and WSAD for Windows 334
Windows/Linux Differences 340

Appendix D. Additional material 343
Locating the Web material 343
Using the Web material 343

Glossary 345

Abbreviations and acronyms 347

Related publications 349
IBM Redbooks 349
Other publications 349
Online resources 350
How to get IBM Redbooks 350

Index 351

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.

Trademarks

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

Approach®
AIX®
CrossWorlds®
CICS®
DataBlade™
Domino™
DB2 Universal Database™
DB2®
ⓔserver™
ⓔserver™
eServer™
Everyplace®
IBM®
ibm.com®
IMS™
Informix®
iSeries™
Lotus Notes®
Lotus®
MQSeries®
Notes®
OS/400®
Redbooks™
Redbooks(logo) ™
RS/6000®
Sametime®
SecureWay®
SupportPac™
Tivoli®
VisualAge®
WebSphere®
z/OS®
zSeries®

The following terms are trademarks of International Business Machines Corporation and Rational Software Corporation, in the United States, other countries or both.

ClearCase®
Rational Software Corporation®
Rational®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation 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.

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.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

Preface

This IBM Redbook is primarily intended for use by IBM Informix Customers and Business Partners. Its purpose is to provide information that will enable the implementation of a robust Web application development environment that can be built with the WebSphere family of products, and based on an IBM Informix Dynamic Server.

The information in this redbook will help you understand the WebSphere family of product packages and Informix Dynamic Server, and how they can be used together. You can even get first-hand experience using the products together. The redbook takes you through the step-by-step process of installing, configuring, using, and managing Informix Dynamic Server Version 9.4 (IDS), WebSphere Application Server Version 5, and WebSphere Studio Application Developer Version 5. And, there are step-by-step instructions that enable you to develop sample applications (using WebSphere Studio) to demonstrate WebSphere and IDS working together in Windows/2000 and Linux operating environments.

In addition to the application exercises, there are many other related topics of interest in this redbook. The following is a brief description of the topics and how this book is organized:

- **The Executive Summary** provides a high level overview of the contents of the redbook. IIt enables the reader to get a basic understanding of the contents and conclusions presented in this redbook, without the requirement of reading all the detailed and technical supporting information.
- **Chapter 1** gives an overview of the Informix family of database products, with a focus on the Informix Dynamic Server (IDS) 9.4. This is a brief overview, and is not intended to be an exhaustive review of IDS. It discusses the basic architecture along with a number of the key functions and features of IDS, with a focus on those used in an integrated environment with WebSphere. You will get a good understanding of the power and capabilities, and the benefits, of using IDS.
- **Chapter 2** guides you through the process of installing and configuring IDS V9.4. Some of the new IDS V9.4 functions and features will be discussed, but the focus in on giving you step-by-step instructions that will help you implement IDS V9.4 on Linux (SuSE V8) and configuring it for use with WebSphere V5.
- **Chapter 3** presents an overview of WebSphere V5 family of product packages. This will be good information for Informix customers who may not yet be familiar with WebSphere. The focus will be on those product packages that will be used in creating a robust application development environment, in conjunction with IDS. There are many product packages in the WebSphere family, but this chapter focuses on those that comprise the WebSphere foundation. And, in particular, WebSphere Application Server and WebSphere Studio Application Developer. Because of their relative importance to this environment, we also discuss WebSphere Portal Server, in Chapter 10, and WebSphere MQ in Chapter 11. Actually, some of the functionality in MQ is used in the sample applications exercises in the form of Java™ Messaging Services.
- **Chapter 4** guides you through the step-by-step process of installing and configuring WebSphere Studio Application Developer. It was installed under Windows/2000, and the focus was on configuration, and integration with IDS.
- **Chapter 5** guides you through the step-by-step process of installing and configuring WebSphere Application Server. It was installed under the SuSE V8.0 Linux operating environment, and the focus was on configuration, and integration with IDS. There is also a related topic covering the WebSphere Administration Repository.

- **Chapter 6** provides more detail on the implementation and integration aspects of WebSphere and IDS. For example, it describes connectivity and integration options, and also alternative implementation scenarios that may be used. The step-by-step process to create a data source on both WebSphere Studio and WebSphere Application Server is given to demonstrate the integration. The result of these exercises is the ability to perform queries from either WebSphere environment and select data from the IDS database.
- **Chapter 7** gets into more detail on working with WebSphere and IDS. For example, it discusses how to create a Java Server Page, how to create a J2EE Container Managed Persistent Bean, and how container managed persistence works. It also takes you through the step-by-step development of sample applications using WebSphere Studio Application Developer. The applications can perform selects, inserts, updates, and deletes against the IDS database. The application makes use of Java Messaging Services, which results in a configuration that could also be used in a distributed systems environment. It also provides guidance for managing transactions.
- **Chapter 8** introduces you to the topic of XML, an industry standard that supports integration and interoperability by enabling data and document interchange. There is an overview of XML and application examples to demonstrate its use. Related topics such as Dynamic XML Mapping are also discussed.
- **Chapter 9** discusses the use of IBM Informix Dynamic Server as a foundation for the implementation of Web services. It describes and defines a Web Services and considers the use of IDS as both a Web Services provider and a Web Services consumer. Examples are provided to demonstrate the use of IDS in this environment.
- **Chapter 10** describes another WebSphere module, the Portal Server. This is an important and strategic element when creating a Web presence. It provides a single point of interaction with dynamic information, applications, processes, and people. This chapter provides an introduction to WebSphere Portal Server and its concepts, and describes how to configure IDS for use with it.
- **Chapter 11** provides a high level overview of WebSphere MQ. Though not directly used in this project, it is another important member of the WebSphere family of products used in support of distributed systems environments. WebSphere MQ can enable integration by helping applications exchange information across multiple heterogeneous platforms. It was used indirectly in this project during the development of our sample application, in the form of Java Message Services.
- **Chapter 12** discusses product directions for an important and widely implemented Informix product, 4GL. It describes the plans for protecting the Informix customer investment in 4GL, extending the functionality, and integrating it into WebSphere Studio.
- **Chapter 13** documents a number of “hints and tips” that are provided to make your implementation of WebSphere and IDS go easier, faster, and more smoothly. These are typically unexpected issues we encountered during the project that we documented so you could be aware of them. Having this information available will save you time and provide the solutions you need during your implementation. We also provide some information regarding other functional areas of importance, such as improving performance.
- **Appendix A** contains additional information about the SQLtoXML and the XMLtoSQL Java classes. These were described and discussed in Chapter 8.
- **Appendix B** provides the complete DADX syntax. It is informational, and was used in the Web Services scenario in Chapter 9.
- **Appendix C** provides an overview of the installation and environment configuration of IDS in a Microsoft® Windows® environment. We focused on Linux in the previous chapters, but wanted to provide instructions for those who may prefer using Windows.

- **Appendix D** provides additional material for this redbook that can be downloaded from the Redbooks Internet site. There are four sample applications, a Sample Applications Implementation Guide, and the Master.css file that is required for the applications, contained there for your use.

Executive summary

This redbook provides, in one document, information you need to understand about how the combination of WebSphere and Informix Dynamic Server can provide you with a powerful and flexible e-business (Web application development) environment. We provide a high level overview of the products for an understanding of the important capabilities that enable such an environment. In addition, we provide detailed and practical guidelines on how to install, configure, integrate, and use these products for development of a robust e-business environment.

Developing a strong presence on the Internet is critical to the success of any business — it is the future unfolding. As such it deserves your attention, and significant consideration in the development of a strategy and direction for developing such a presence. It is too important to treat lightly or to believe you have multiple opportunities, or that the competition will wait for you to “get it right”.

IBM is a leader in providing products that can help enable a robust e-business environment. They have invested heavily in research, and development of solutions to address it. The result is a strategic infrastructure software platform for dynamic e-business, called WebSphere. This is not just a product, but a comprehensive family of products designed on open standards. It has been designed for easy implementation and use, but with the flexibility you need as you grow and change.

IBM is also a leader in delivering high volume and high function data management products, and relational database management systems in particular. With the acquisition of Informix, IBM has gained database market share and the Informix portfolio of data management products. The Informix products are also flexible and robust, and have been serving the Informix customer set for a long period of time. Becoming part of the IBM family, the Informix customers now have even better access to, and incentive to use, other IBM products. And, in particular, WebSphere.

You should read this redbook and consider the benefits of developing your e-business environment on these product platforms. It is written to help you easily implement the products and demonstrate for yourself the power and capabilities. We take you through step-by-step product installation processes, and then through step-by-step development process for some sample applications. This will give you a good basis for evaluating these products, their ease of installation, their ease of integration, their ease of enabling application development, and their usability. It is a practical approach that lets you see, do, and discover for yourself, rather than simply telling you about it, or just showing you an example. Take the time to try it and evaluate it, and we think you will choose it.

The sample applications

To make this a more practical decision process, we decided to enable you to install the required products, configure them to work together, and then develop some sample applications to validate the integration and the ease of use of the application development environment. The sample applications discussed in the redbook are based on several application development approaches. That is, they demonstrate flexibility in the type of development and in the type of systems environments. For example, they describe application development for both a single server environment and a distributed systems environment. Since you will be using real products and developing your own sample applications, you can also try your own preferred approach.

Highlights and benefits

A number of highlights and benefits are described and demonstrated in the redbook. For quick and easy reading, we have simply provided some of them in the following list form. For more details, and to add to the list, we encourage you to continue reading the remaining content of the redbook.

- Informix Dynamic Server is a robust and powerful relational database management system. It has the scalability, reliability, and availability to support the needs of today's high volume and high availability e-business environments.
- WebSphere is a robust and powerful software platform for e-business. It provides products to provide all the capabilities required for an e-business environment. It has been architected for the development, deployment, management, and use of dynamic e-business applications.
- WebSphere Studio Application Developer provides the capability to develop and deploy high function e-business applications. It includes a graphical oriented development approach that fast delivery and minimum resource requirements.
- WebSphere Application Server provides the capability to manage the deployed e-business applications. It enables the end user connectivity, execution, and management of the e-business workload.
- Step-by-step guidelines for the installation, configuration, and integration of the e-business products enables the fast implementation of a sample e-business environment. They provides a practical and fast approach for trying and evaluating the e-business capabilities that can be provided by an implementation of the products considered in this redbook.
- Information and examples on advanced topics, such as XML, provide education, guidance, and direction on how to use and position products based on this powerful technology. Based on industry standards, this is an important technology for supporting integration and interoperability by enabling data and document interchange in heterogeneous environments.
- New technologies are presented that can enhance the development capabilities, such as Web Services. A Web Service is a set of related application functions that can be programmatically invoked over the Internet. Businesses can dynamically mix and match Web Services to perform complex transactions with minimal programming. This means less resources for, and faster development of, applications

- WebSphere Portal Server is discussed and positioned as a single point of interaction with dynamic information, applications, processes, and people to help build successful business Web portals. This is critical to the ease of learning and ease of use of your e-business environment. It supports a wide variety of pervasive devices enabling users to interact with their portal anytime, anywhere, and using any device, wired or wireless.
- Any e-business environment will be required to handle thousands of transactions to enable users to exchange information across many different platforms. They will be sending and receiving data, as messages, on a continuous basis. Your system will need to connect all your business software together to form one efficient enterprise by providing an open, scalable, industrial-strength messaging backbone. This is what WebSphere MQ does. It can minimize the time taken to integrated resources and applications held in different systems so you can quickly respond to the changing needs of the e-business.
- Many Informix customers use the Informix 4GL product for application development. We provide information on how IBM is continuing to enhance this product and protect your investment. As such, the direction is to integrate that product into the WebSphere family through integration into the WebSphere development environment.
- We also provide a number of Hints and Tips to make it easier as you implement your e-business environment using IDS and WebSphere.

The redbook provides, in a single document, the information you need to implement and integrate WebSphere and IDS. And, it guides you through the development and deployment of sample applications to demonstrate and verify the integration.

We hope you find the information in this redbook informative, educational, practical, and useful in developing an example integrated e-business environment with WebSphere and Informix Dynamic Server.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world. Four members worked on this project at the International Technical Support Organization, in San Jose, California and one worked remotely in Boston, Massachusetts.

Left to Right: Pagadala J. Suresh, Eduardo Akisue, Chuck Ballard, Alexander Koerner

Chuck Ballard is a Project Leader at the International Technical Support Organization, in San Jose, California. He has over 35 years experience, holding positions in the areas of Product Engineering, Sales, Marketing, Technical Support, and Management. His expertise is primarily in the areas of database, data management, data warehousing, business intelligence, and process re-engineering. He has written extensively on these subjects, taught classes, and presented at conferences and seminars worldwide. Chuck has both a Bachelors degree and a Masters degree in Industrial Engineering from Purdue University.

Eduardo Akisue is an Advanced Technical Support Engineer at the IBM Latin America Call Center, located in Miami Florida, USA. Originally joining Informix in 1996, he has worked with almost every product in the Informix portfolio. Now with IBM, he is an Informix and DB2 certified engineer and also holds the Online III certification (dial-up certification for down systems cases). His current focus is on IBM Informix Dynamic Server, DataBlades, and Content Management, and activities include technical support to other engineers, presentations, and classes through Latin America countries, and advanced support for critical situations. He provides customer support in three different languages (English, Portuguese, and Spanish), and holds a Bachelors degree in Computer Science from the University of Sao Paulo - Brazil.

Alexander Koerner is an Informix-, DB2- and XML-certified Senior IT-Specialist in the Data Management Technical Sales organization, Munich, Germany, and joined Informix in October 1989. He was instrumental in starting and leading the SAP/R3 on Informix project, developed an Informix adaptor to Apple's (NeXT's) Enterprise Object Framework, and has contributed to many strategic projects across the region. Alexander is currently focused on topics such as IDS 9, XML, Web services, and DataBlade™ technology. His activities also include presentations at conferences such as the IBM Software Symposium, IIUC, XML One, and ApacheCon. He is a member of the German Informatics Society and holds a Masters degree in Computer Science from the Technical University of Berlin.

Pagadala J. Suresh is a Software Engineer, working in India, with two years of experience in the area of Web Technologies. He holds a Masters degree in Computer Applications from Karnataka Regional Engineering College, Surathkal. His areas of expertise include WebSphere Application Server and WebSphere Studio Application Developer (WSAD), and he is a certified Developer Associate on WSAD. He has written extensively and has recently submitted a paper to the IBM developer works, and has given numerous presentations on these subjects. In addition, he has hands-on experience with the WebSphere Business Components Composer framework by IBM, and has had an asset on Testing Strategy - Static and Dynamic accepted and posted on the IBM Intellectual Capital Management Asset Web.

Martin Lurie was the remote team member. Marty is a Systems Engineer in IBM's Data Management Division, and is located in Boston Massachusetts. However, if pressed, he will admit that he mostly plays with computers. He is an IBM-certified DB2 DBA, IBM-certified Business Intelligence Solutions Professional, and an Informix-certified Professional.

Extended team members

We would like to give special thanks to the following people. Although not formal members of the residency team, they contributed advice, support, and written content, and were considered to be “extended team members”.

Jonathan Leffler is an IBM Informix Database Engineer in Menlo Park, California. One of his areas of specialty is the Informix 4GL.

Sekhar Meka is an Application Architect providing e-business Architecture Services, from AMS e-bis. His areas of specialty are WebSphere Application Server, WebSphere MQ, WebSphere Portal Server, and WebSphere Pervasive Computing products.

Allan Richen is an IT Specialist providing e-business Enablement Services in Winnipeg, Canada. He is an IBM-certified specialist for WebSphere Application Server and Java 2, and the WebSphere Competency Lead for the Canadian SDC.

Other contributors

We would also like to thank the following people for their support and contributions:

Omkar Nimbalkar, Informix Product Management and Marketing
Dwaine Snow, Product Manager, Informix Dynamic Server
Patricia Quinn, Brand Manager, Informix Dynamic Server
Mohan Natraj, WW Brand Manager, XPS and RedBrick
Helen Wong, IDS 9.4 Development Manager
Sonali Surange, DADX Web Services
Nelson Androes, JDBC
Fred Summa, JDBC
Gary Proctor and Team, J/Foundation
Vinayak Shenoi, J/Foundation
Martin Siegenthaler, MQ DataBlade
Sandor Szabo, Linux
IBM Informix Marketing, Support, and Development

Susan Malaika, DB2 XML
Dirk Wollscheid, WORF
IBM DB2 Development

Janet Olausen, WebSphere Technical Support Marketing Program Manager
Glenn McGorty,, Manager, WebSphere Trial Support
David Long, WebSphere Software Developer and Program Support Team
Ernest Mah, WebSphere Application Developer
Kihup Boo, WebSphere Application Developer
Dirk Wittkopp, WebSphere Portal Server
Uwe Hansmann, WebSphere Portal Server
Muthukumarasamy L Shanmugam, WebSphere Application Server
IBM WebSphere Development

Jaques Roy, WorldWide Sales Support
DB Server Technology Sales

Mary Comianos, Operations and Communications
Yvonne Lyon, Technical Editor
Deanna Polm, Residency Administration
Emma Jacobs, Graphics Designer
International Technical Support Organization, San Jose Center

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 Internet note to:

  `redbook@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. QXXE Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099

Introduction

Because the world is rapidly moving more and more business to the Internet, developing an Internet presence will most certainly have a very positive impact on the success of almost any business. Many businesses have been on the Internet for some time and have gone through a number of iterations as they determine what works best for them. Key to this is the ability to develop and deploy Web applications easily and quickly, as well as to maintain their flexibility to change and adapt.

The changing business environment

Flexibility, and easy and quick deployment of Web-based applications, has really become more than just important for businesses, it has become an imperative. It is becoming even more important for businesses, with the increasing demand for consistent quarterly profits and growth. Stockholders and financial analysts are keeping an ever more watchful eye on these targets, and becoming less and less forgiving when they are not met. It is certainly a changing business environment, and one that demands businesses manage to it. To meet this challenge means becoming more efficient and more effective. Everyone is being asked to do more with less — and the competition is fierce.

An element of every business strategy is, and must be, an increasing use of the Internet. It is fast moving and ever changing, and requires maximum flexibility. To satisfy this requirement, businesses are turning to their IT organizations. They look for faster and easier-to-use applications that are easily modified and expanded. All this in an environment of mergers and acquisitions that require integration of disparate systems, faster changing user requirements, and the need for more information faster. This is also mandating that IT shops begin to purchase more software rather than build it, and that they use products based on open standards to minimize development and support requirements.

In turn, IT shops are demanding that software developers deliver products that support the integration of their environments, which typically consists of products from multiple vendors. And, they are beginning to base purchasing decisions on the ability of products to participate in an integrated environment. Many times this simply means architecting a framework based on a qualified set of vendors and a preferred technology, and then building on it. Gone are the days when IT shops could afford to support multiple products that provide similar capability. They now must consider consolidation, standardization, and commoditization. These are the types of things that can keep overhead low, expenses low, and prices low, to help meet those quarterly targets.

IBM® supports those goals by delivering products built on open architectures and with open standards. And this is the case with the WebSphere® family of products. They provide a common application development environment that is easy to implement and use, that supports standards, and that can operate in an integrated environment. This enables a powerful and high function application development environment.

Why Informix® and WebSphere?

With the advent of the Internet and e-business, the demands on the IT infrastructure are greater than ever, from both a performance and data management perspective. Also critical, are the applications that help meet the consumers business demands. The DB2® and Informix database products are very powerful, stable, and proven, and provide customers with a very robust infrastructure upon which applications can be built. And, WebSphere is there to meet the application development requirements.

Now that Informix has been acquired by, and integrated into, IBM — Informix customers also have an excellent opportunity to take advantage of this robust and flexible Web application development environment much more easily. WebSphere can enable Informix customers to take greater advantage of the e-business environment, and help meet their current Web application development needs today — as well as their growing and changing requirements of tomorrow.

Using Informix and WebSphere

In this redbook we present information to help in the installation, configuration, use, and management of a Web applications development environment consisting of IBM WebSphere V5 and Informix Dynamic Server V9.4. It will help enable customers to take advantage of the many benefits and productivity gains that come with using the WebSphere open, common development toolset. This redbook is primarily written for Informix customers and business partners who may not yet be familiar with WebSphere, and those who have already selected WebSphere and are in the process of implementing one or more of the WebSphere product packages. Much of the information in the book was obtained from experience gained by the redbook project team from installing the products, using WebSphere to develop sample applications, and running those sample applications using IDS as the underlying relational database.

The redbook describes some of the functions and features of Informix Dynamic Server, WebSphere Application Server, and WebSphere Studio Application Developer that support the challenges of the changing business environment. Informix Dynamic Server (IDS) is a seasoned relational database management system supporting many mission critical applications, and enjoys a very large customer install base. WebSphere is the IBM strategic software platform for e-business and supporting a number of underlying databases — including IDS. Now Informix customers can take advantage of this powerful e-business platform for their Web application development.

We provide a brief overview of both the Informix Dynamic Server and the WebSphere family, to give you a base from which to start. This information will give a better understanding of the products and how they can work together to enable you to get the benefits of, for example, a powerful Internet presence much more quickly.

To help you get you started towards such an environment, we implemented the products ourselves so we could provide you with some guidelines and instructions to make your implementation easier and faster. For our implementation, we used an intel-based IBM processor, with 760 MB of memory, and the following software configuration:

- SuSE Linux V8.0
- Informix IDS V9.4
- WebSphere Application Server V5 for Linux

Sample applications were developed using WebSphere Studio Application Developer Version 5, running under Windows/2000. A sample database, the Stores_demo Database, comes with IDS and was used as the test vehicle for the implementation. It is a database familiar to most Informix customers. This environment provided us a means of demonstrating how quickly applications can be developed with WebSphere to retrieve data from IDS databases. Windows/2000 based clients were used to access the WebSphere Application Server, the sample applications, and IDS.

In addition, we provide overviews of other members of the WebSphere family, such as Portal Server and WebSphere MQ, even though they were not part of our actual implementation. We also discuss other subjects such as Web Services and XML, and position them as they relate to an integrated IDS and WebSphere environment. In addition, we provide information on the future direction for the Informix 4GL and how it fits in with the WebSphere strategy.

We believe this redbook will provide, in one document, the information you need to begin implementing a powerful Web presence for your organization — based on robust, market leading products. And, that Web presence will be such that it provides the flexibility you need as you move forward, grow, and change.

Using our sample applications

Note: We created and documented, in this redbook, the exercise of developing sample applications that enable you to get first-hand experience at using the real IBM products working together. However, this is not a simple task. To do these exercises requires the installation of IDS 9.4, WebSphere Studio Application Developer, and WebSphere Application Server, along with a few related components that we provide. If you do not already have these products, you can download trial copies of them from the IBM Informix Web site and the IBM WebSphere Web site.

The step-by-step instructions for developing the sample applications are documented throughout this redbook. However, we have also created a “Sample Applications Implementation Guide” to put all those instructions in one place to make it simpler for you. That guide can be downloaded from the Redbooks™ Web site. Instructions can be found in “Additional material” on page 343.

There may be those who prefer only to implement the software and then execute the sample applications. To do this, you can download executable copies of the sample applications. They are actually included in the same download file discussed above with the Sample Applications Implementation Guide.

1

Informix Dynamic Server: An overview

In this chapter we provide a brief overview of the Informix Dynamic Server (IDS). This is not intended to be an in-depth product description, nor does it describe all of the functions and features of IDS. Rather, it presents some of the basic concepts, architecture, functions, and features that contribute to the powerful capabilities of the product.

Many of the features discussed are centered around the ability to support high availability, fast data loads, fast backup and restore, and high performance. These are the types of capabilities required to develop and support a robust online environment, and still maintain flexibility and ease of change. And, this is the type of environment that can be created as you implement IDS in an integrated environment with WebSphere.

1.1 Informix database family

In this section we provide a very brief description of the Informix family of database servers. We describe them briefly, and position the features and advantages that enable them to satisfy the differing requirements of the solutions that customers need.

Informix Dynamic Server (IDS)

IDS is the Informix flagship database server. It is the best-of-breed, general purpose, but mission-critical, OLTP database for e-business. It delivers the performance, reliability, scalability, and high-availability needed for today's global, e-business enterprises.

Key IDS features include Data Partitioning, High-Availability Data Replication (HDR), Enterprise Replication (ER), Parallel Queries, and fast loads and backups. It also supports industry standards for client connectivity, such as ODBC, JDBC and OLE/DB.

IDS is divided in two main product lines: Version 7.x and Version 9.x. The IDS architecture, and most of the features, are supported on both families. Some extended features for Version 9.x will also be described in later chapters.

Informix Dynamic Server with J/Foundation

IDS with J/Foundation combines all the features of IDS with an open, flexible, embedded Java Virtual Machine. Basically this provides the added capability of running Stored Procedures written in the Java language (Java on the server).

Informix Online Extended Edition

Informix Online Extended Edition is an easy-to-use, relational database server for low-to-medium size workloads. It features superior online transaction processing support with the assurance of data integrity. And it provides rich multimedia data management capabilities that support the storage of a wide range of media, such as documents, images, and audio. Included also is support for a wide variety of application development tools, along with a large number of other third-party tools, enabled by support of the ODBC and JDBC industry standards for client connectivity.

Informix SE

Informix Standard Edition (SE) is a UNIX, Linux, and Microsoft Windows-based embeddable database server. It seamlessly integrates with Informix and third party application development tools, and it is ideal for small to medium-sized applications that need the power of SQL without any database administration requirements. It represents a low-maintenance, high-reliability database solution, and it provides excellent performance, data consistency, client/server capabilities, and adherence to standards.

Informix C-ISAM

Informix C-ISAM is a fast and cost-effective file management technology. It is a library of C functions which efficiently manage your indexed sequential access method (ISAM) files. As it is only a file management system (and not a database), it eliminates the overhead of a full relational database management system (RDBMS). The result can be fast, direct access to your data records.

Informix Extended Parallel Server (XPS)

Informix Extended Parallel Server (XPS) is designed for the largest, most demanding, and complex data warehouse applications. It provides comprehensive, but flexible, data warehousing features enabling integration of business systems through a sophisticated shared-nothing underlying architecture. It delivers industry-leading performance, flexibility, and scalability, and enables fast, informed decision making from complex, query intensive analytical applications.

1.2 Understanding Informix Dynamic Server (IDS)

The database focus of this book is on the IDS product. This section provides an overview of the IDS functions and features that contribute to the value of integration with WebSphere.

IDS is a fast and powerful OLTP oriented relational database management system. It is based on the multi-threaded Dynamic Scalable Architecture (DSA) developed by Informix. This architecture requires fewer processes to take care of multiple database activities. That means that as more users are added to the system, we can dynamically add the necessary resources the server needs to manage those user transactions. DSA was designed to provide efficient resource utilization so you need less hardware to support your growing business needs. It also uses object relational technology (but only in Version 9) that can dramatically increase the performance and efficiency of enterprise applications.

IDS architecture

The IDS architecture can be described in terms of three components:

- Multi threaded implementation
- Database server
- Client-server connectivity

Next we briefly describe these components and explain how they are related.

Multi threaded Implementation

IDS uses a multi threaded architecture. To better explain this implementation, we should first understand the concept of a thread. A thread is a set of instructions that are executed in a program. When many threads are running in a single process, it is called multithreading. A program that does not implement support for threads is called a single threaded process.

A multi threaded process can run multiple threads in a single process. Each thread runs sequentially and gives up control to another thread at a specific point in time. Each process is initiated by the operating system (OS), which takes time and resources. Multi threading then provides a method of executing many iterations of a process for different users, without having to incur the overhead of instantiating new OS processes. So, minimizing the number of processes that must be started, controlled, and ended, saves time and resources — with the result being improved performance.

See Figure 1-1 for a comparison of single threaded and multi threaded processes.

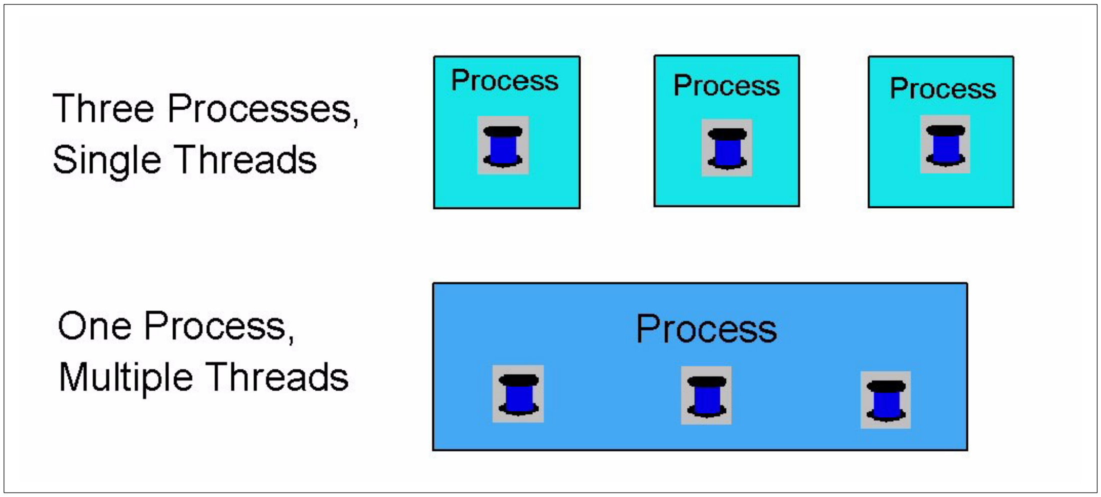


Figure 1-1 Single threaded process versus Multi threaded process

Database server

The database server consists of three major components: processes, disk, and shared memory. This is depicted in Figure 1-2. Each of these components plays an important part in determining the overall server performance, which includes workload capacity, response time, availability, reliability, and resource consumption.

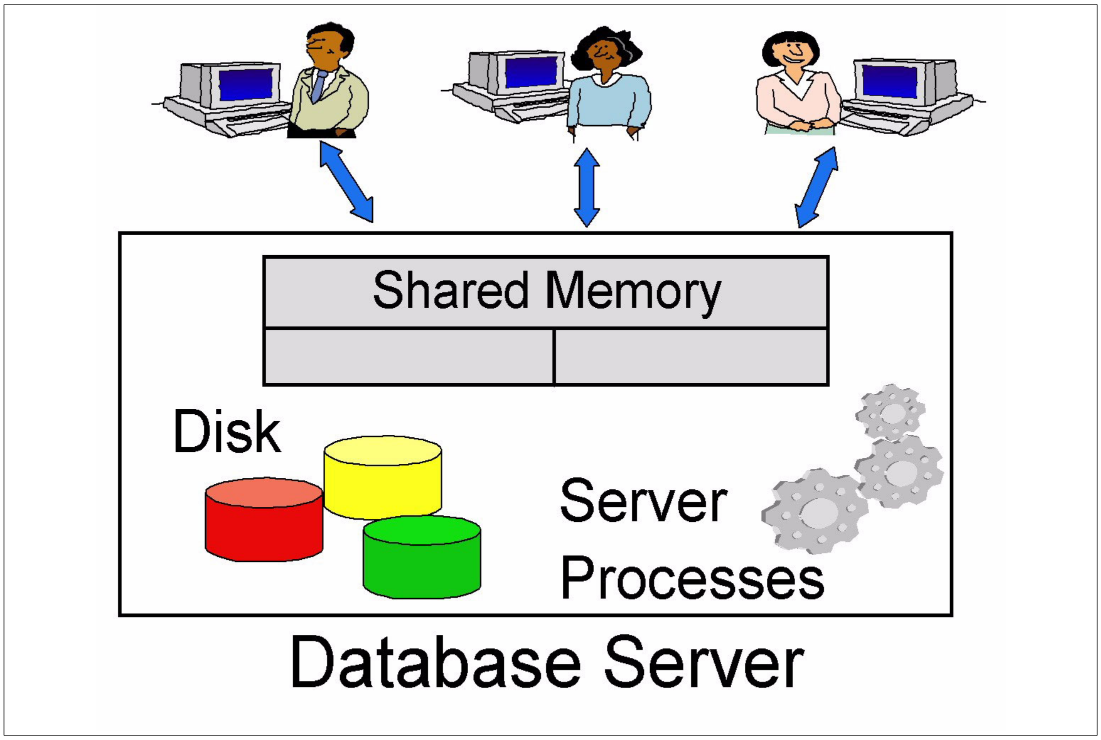


Figure 1-2 IDS database server

- **Process Component:** The Process Component consists of a set of *Virtual Processors* (VPs). They are called Virtual Processors because they process and schedule the activities of their own threads. At the OS level each of these processes have the name of *oninit*. A VP belongs to a VP class that is responsible for specific tasks in the database server. The most common VP classes are:
  - **CPU:** The Central Processing Unit is where all processing activity is performed. All user threads, and some system threads, run on CPU class VPs. The CPU processes are usually very busy and attempt to fully exploit the processing cycles granted by the Operating System. The administrator can increase or decrease the number of CPU processes as needed with the server in online mode. This dynamic resource management capability is one of cornerstones of DSA.
  - **AIO:** Asynchronous Input/Output processes are responsible for executing I/O operations to disk. AIO processes also perform all I/O to server associated files, such as the IDS log message file. They are automatically configured when the server is initialized, but more can be dynamically configured while the server is running.
  - **PIO:** Physical Input/Output runs internal threads to perform I/O on a special disk component called *Physical Log*. The Physical Log is a collection of contiguous disk pages and it is used for recovery purposes. It does this by storing *before-images* (first copies) of data pages that have been modified in shared memory
  - **LIO:** Logical Input/Output runs internal threads to perform I/O on a disk component called *Logical Log*. Logical Log files are collections of contiguous pages used for transaction records. They are used for logical recovery of databases with *logging* transactions enabled.
  - **ADT:** Runs threads for auditing purposes.
  - **MSC:** Performs miscellaneous tasks, such as user authentication.
  - **ADM:** Handles the timer that is used for activities that are scheduled to run for a certain period of time. For instance, some threads go to sleep for a certain number of seconds. It is the timer’s responsibility to mark the time of these threads.
- **Shared Memory Component:** The Shared Memory Component in the database server is divided in three parts, that are described below:
  - **Resident:** The resident part contains the buffer pool and other system information. The buffer pool holds pages from database tables and consumes the largest amount of memory. It can be configured to remain resident in the main memory if this feature is supported by the Operating System.
  - **Virtual:** The virtual part contains the information about the threads and sessions, and the data used by them. This portion grows and shrinks frequently so it is the database server responsibility to allocate or deallocate memory for this part.
  - **Message:** This part holds the buffers that are used in communication between the clients and the server when the communication method is configured as shared memory.
- **Disk Component:** The database server requires disk space to store data. This space is assigned through physical and logical units; the primary units are described below:
  - **Chunks:** A chunk is a contiguous unit of space that is assigned to the server. The server manages the use of space within that chunk.
  - P**ages:** A chunk is comprised of units called pages, which are the smallest unit of I/O for the database server. All database and system information is stored in pages.
  - **Extents:** These are defined contiguous collections of disk pages, that get assigned to applications for use.

  - **Dbspaces:** A dbspace is a logical collection of chunks, and it must have at least one chunk assigned to it.
  - **Tablespaces:** A tablespace is a logical collection of extents and can be thought of essentially as a database table. It contains all pages that contain data or indexes for the table.
  - **Blobspaces:** A blobspace is a special type of Dbspace. It is created to store special data types called large objects. There are two types of large objects: TEXT and BYTE. The TEXT data type is used to store large amounts of ASCII text, such as source code and scripts. BYTE is used to store any kind of binary data, such as digitized images or sound.
  - **Sbspaces:** An sbspace is also a special type of dbspace, for storing smart large objects. Smart large objects include built-in data types, such as Character Large OBjects (CLOBs) and Binary Large OBjects (BLOBs), third-party customized indexes, and User-Defined Data Types (UDTs).
  - **Physical Log:** The server has a special log that is used for automatic recovery purposes, called the physical log. The physical log is a collection of contiguous pages on disk. When a page is read into a shared memory buffer and modified by a user, a copy of the page in its original condition is written to the physical log. This copy of the page is known as a before image (the copy of the page before it was changed). Only the first change to a page in a buffer causes a before image to be written to the physical log. Any subsequent changes to that same page do not cause additional before images to be written to the physical log. These before images are used by an automatic recovery mechanism.
  - **Logical Log:** The logical-log files are collections of contiguous pages on disk that are used to store transaction records for the server. These transaction records are used to track all the changes made to databases that were created with logging. All databases share the same set of logical-log files, and each server must have at least three logical-log files.

Client-server connectivity

There are four methods for a client application to connect to the database server:

- **Shared memory:** When the client application and the database server are on the same host computer, this is the preferred method of communication. The client application and the server attach to the same segment of shared memory. The application leaves messages for the server and picks up messages left for the server using this memory.
- **Stream pipes:** This is a local inter-process communication protocol that uses UNIX streams. This is a UNIX specific method.
- **TCP/IP:** Using the TCP/IP protocol a client application can connect to either a local or remote database server. There are two types of TCP/IP implementations: Socket and Transport Layer Interface (TLI).
- **IPX/SPX:** This is a specific protocol to connect to NetWare Servers.

When an application attempts a connection to a database server, some basic information is needed to enable the connection. This information is held, by default, in a file called $INFORMIXDIR/etc/sqlhosts, and must be maintained by the system administrator. The user running the application must set the INFORMIXSERVER environment variable to a key name that points to the entry in sqlhosts. For a server connection, the key name is the same value as the DBSERVERNAME or DBSERVERALIASES parameter found in the configuration file (onconfig). The key name can be any unique name within the sqlhosts file. Once the application finds the correct entry in the sqlhosts file, it has the information necessary to connect to the database server.

The sqlhosts file has four main columns that follows the format illustrated in Figure 1-3.

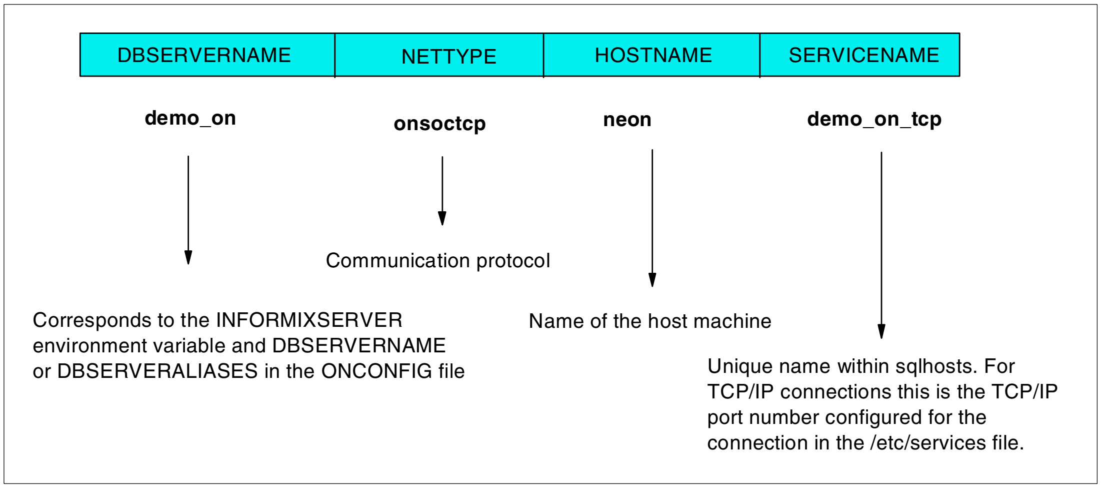

DBSERVERNAME
NETTYPE
HOSTNAME
SERVICENAME
demo_on
onsoctcp
neon
demo_on_tcp
Communication protocol
Corresponds to the INFORMIXSERVER environment variable and DBSERVERNAME or DBSERVERALIASES in the ONCONFIG file
Name of the host machine
Unique name within sqlhosts. For TCP/IP connections this is the TCP/IP port number configured for the connection in the /etc/services file.

Figure 1-3 SQLHOSTS file

To better understand how the client application environment variables, the database server configuration file (onconfig) and the sqlhosts relate, see Figure 1-4, which has actual values in all the columns.

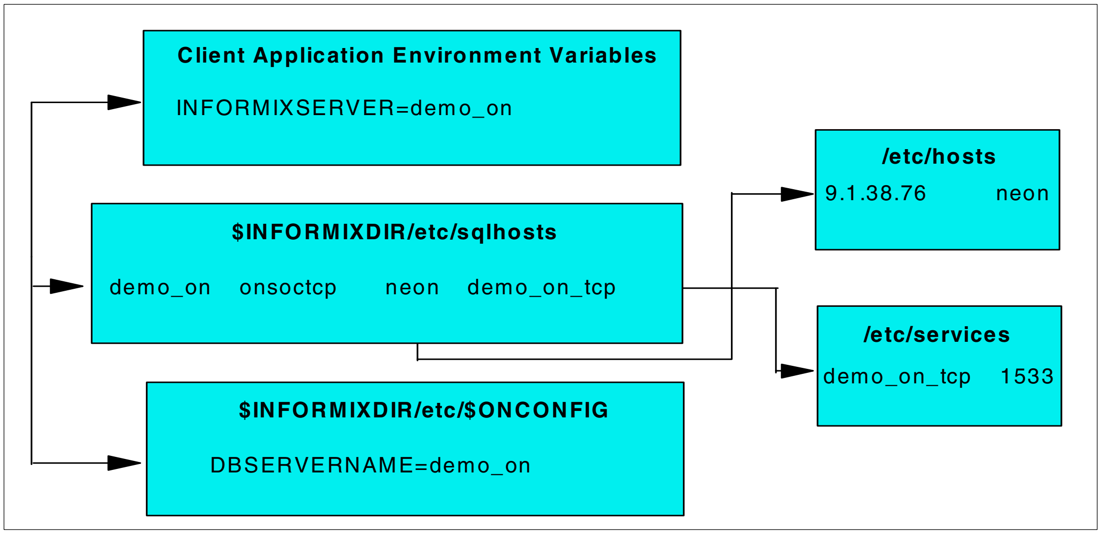
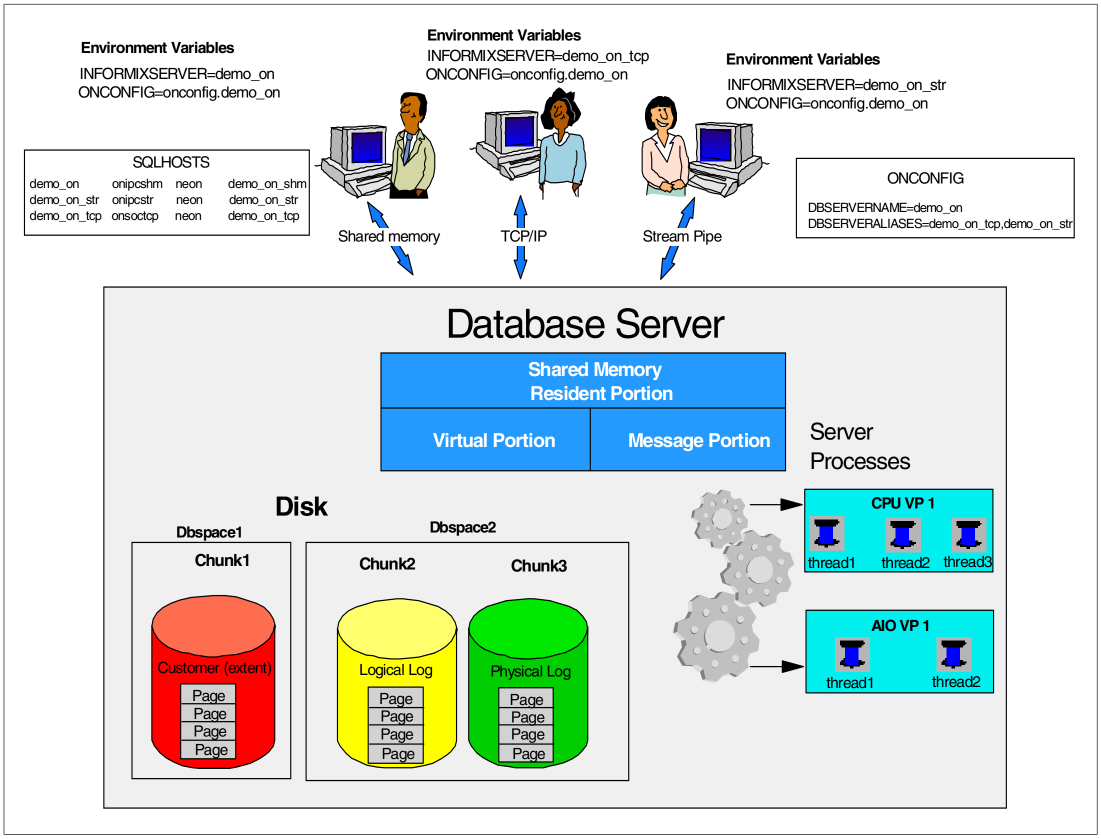

Client Application Environment Variables
INFORMIXSERVER=demo_on
/etc/hosts
9.1.38.76 neon
$INFORMIXDIR/etc/sqlhosts
demo_on onsoctcp neon demo_on_tcp
/etc/services
demo_on_tcp 1533
$INFORMIXDIR/etc/$ONCONFIG
DBSERVERNAME=demo_on

Figure 1-4 Relationship between sqlhosts, onconfig and environment variables

The “Big Picture”

We can now modify Figure 1-2 on page 4 based on the information and concepts described in the previous sections regarding the IDS architecture. This will give us the “Big Picture” of how it all relates. This is depicted in Figure 1-5.



```
SQLHOSTS
demo_on       onipcshm   neon      demo_on_shm
demo_on_str   onipcstr   neon      demo_on_str
demo_on_tcp   onsoctcp   neon      demo_on_tcp
```

```
ONCONFIG
DBSERVERNAME=demo_on
DBSERVERALIASES=demo_on_tcp,demo_on_str
```

Figure 1-5 The “Big Picture” of IDS

1.3 Feature highlights

This section describes some of the IDS features and explain the benefits of each for an Enterprise System. IDS has some important features that provide better performance, enables server high availability, and improves reliability.

Many of the features discussed here are valid for both IDS Version 7 and 9, and some of the extended features for Version 9 are also presented. Specifically, we will focus on versions 7.31 and 9.30. In Chapter 2, “Installing and configuring IDS V9.40” on page 21, we describe some of the new features in IDS 9.40, the latest IDS version that is now available.

Table and index partitioning (fragmentation)

Fragmentation is the distribution of data from one table across many dbspaces. What this means is that even though the table is distributed in many dbspaces, it is still treated as a single table by SQL. This is depicted in Figure 1-6.

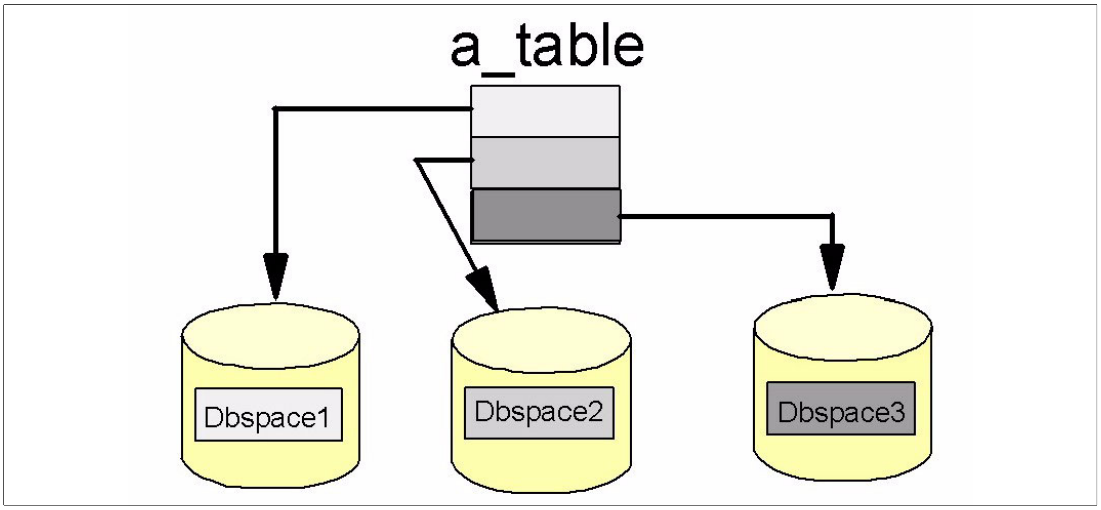


Figure 1-6 Table fragmentation

The main advantages of table and index partitioning are as follows:

- **Parallel scans and other parallel operations:** When the database server is configured for Decision Support System (DSS) environments, we can enable a feature called Parallel Data Query (PDQ). With this feature we get parallel fragment scans. DSS systems get a significant performance advantage with parallel scans, particularly since their queries typically need to read large volumes of data.
- **Balanced I/O:** Fragmentation of tables also balances the I/O across multiple disks, meaning that we avoid disk contention and bottlenecks. This is particularly good for On-Line Transaction Processing systems (OLTP) where a high throughput is critical.
- **Archive and Restore:** Fragmentation provides a finer granularity in terms of backup and restore. Since the table is split in different dbspaces, the backup or restore can be performed at the fragment level.
- **Higher Availability:** You can configure whether the server should skip unavailable fragments. This is especially advantageous in DSS systems, when a high volume query has been running for a long time and then encounters a fragment that is not available. The query would have taken significant time and resources, but would no have completed.

High-Availability Data Replication (HDR)

HDR provides synchronous and asynchronous replication of an entire database server (instance) on different servers that, in some cases, can be located in different sites. The advantages of HDR include:

- **Performance:** Clients at the site where the database server was replicated experience improved performance, since they can access the database locally.
- **High Availability:** Clients at all sites experience higher availability of the data. Even if the local system is unavailable they can still connect to the remote database server.

In an HDR environment, we have two servers involved: one is called the *primary* database server and the other *secondary* database server. In this high-availability context, a database server that does not implement HDR is called a *standard* database server. During normal operation, clients can connect to the primary server and use it as they typically would. They can also use the secondary server, but only for read activity. The secondary server does not permit update operations.

As illustrated in Figure 1-7, the secondary database server is dynamically updated.

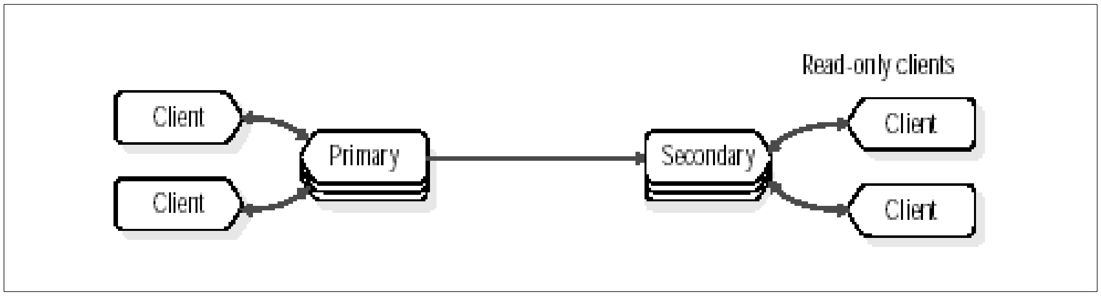


Figure 1-7 Primary and secondary server in an HDR implementation

If the primary server fails, you can change the secondary server to a standard database server and redirect all client connections, as illustrated in Figure 1-8.

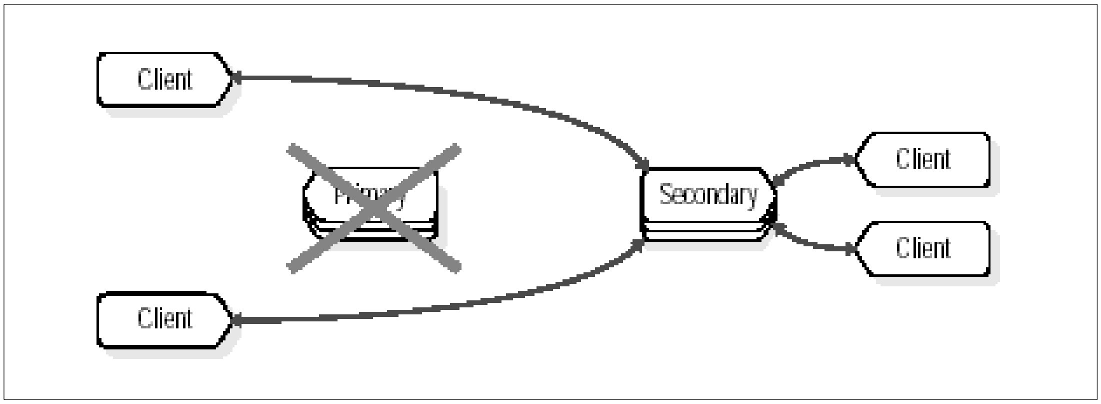


Figure 1-8 HDR servers and clients after a failure

Clients connect to an HDR server the same way they would connect to a standard database server. In a case of a failure they might be redirect to the secondary server, so they continue to be fully operational. The database server does not provide a transparent way to redirect client applications to the secondary server, but there may be other ways to automate this process:

- **Automatic Redirection with DBPATH:** The DBPATH redirection method relies on the fact that when an application does not explicitly specify a database server in the CONNECT statement, and the database server specified by the INFORMIXSERVER environment variable (variable used to specify the database server for client connections) is unavailable, the client uses the DBPATH environment variable to locate the database (and database server).
- **Administrator-controlled redirection with connectivity information:** Clients can be redirected by the server administrator. The connectivity configuration file (sqlhosts) can be used to make the necessary changes for the clients.
- **User-controlled redirection with INFORMIXSERVER:** When an application does not specify the database server name in the CONNECT statement, the connection is performed to the value specified in the INFORMIXSERVER environment variable. In case of a failure in an HDR environment, applications that use that database server can reset their INFORMIXSERVER variable to the other server in the HDR pair.

For more information about HDR, see the *IBM Informix Dynamic Server Administrator's Guide, Version 9.4*, G251-1249-00.

Enterprise Replication (ER)

Enterprise Replication implements asynchronous data replication based on using log data capture as the source for the replication.

Asynchronous replication means that updates on databases that reside at a replicated site occur after the primary database has committed the changes. The delay to update the replicated-site databases can vary depending on the business application and user requirements. However, the data eventually synchronizes to the same value at all sites. The major benefit of this type of data replication is that if a particular database server fails, the replication process can continue, and the original transaction is not directly affected by replication.

Enterprise Replication uses a log-based transaction capture mechanism to capture executed transactions. A log-based transaction capture minimizes the impact on transaction processing performance because it captures changes from the log rather than competing with transactions that access production tables. Enterprise Replication reads the logical log to obtain the row images for tables that participate in a replication process. Enterprise Replication marks these rows when your transactions are logged. These rows are then passed to Enterprise Replication for further evaluation.

Enterprise Replication supports the following replication models:

- **Primary-Target:** Unidirectional updates from a primary database server to many target database servers, or from many target database servers to a primary database server.
- **Workflow:** A database is updated at one location, then the updates are passed to another location for update, then passed to another location for update, and so on.
- **Update-Anywhere:** All databases have read and write capabilities. Updates are applied at all databases.

In order to use ER, we first need to configure the database servers for replication (Enterprise Replication server) and then create replicates, which are the ER objects that replicate data. The replicates contain information such as replication type, participant ER servers, database name, table owner, table name, and SELECT statement. The replicates are defined at a table level and use a SELECT statement to specify which columns or rows of this table are to be replicated among the ER participant servers.

The primary difference between ER and HDR is that HDR replicates the entire instance to a remote site, while in ER the replicates are defined at a table level and support different models to be implemented (Primary-Target, Workflow, and Update-Anywhere). Also, HDR supports both synchronous and asynchronous data replication while ER only supports the asynchronous mode.

For more detailed information about Enterprise Replication, see the *IBM Informix Dynamic Server Enterprise Replication Guide, Version 9.4*, G251-1254-00.

Parallel backup and restore

IDS performs parallel backup and restore through ON-Bar, which is one of its Backup and Restore utilities. The other is ontape, but it only performs backup and restore serially.

ON-Bar can be used to make a backup copy of your database server data and logical logs to be used as insurance against lost or corrupted data. Data might be lost or corrupted for reasons that range from a program error to a disk failure to a disaster that damages the facility in which your computer resides. To recover data, restore the database in two steps: First restore the backup copy of the data and then restore the logical logs to bring data as close as possible to the most recent state.

ON-Bar works in conjunction with another software layer called Storage Manager and they communicate to each other using the X/Open Backup Services Application Programmer's Interface (XBSA). IDS comes with a simple Storage Manager called Informix Storage Manager (ISM), but ON-Bar can also work along with IBM Tivoli® Storage Manager (TSM) and also other third-party Storage Managers, such as HP Omniback.

The ON-Bar components include:

- Both onbar and onbar_d — onbar is a script shell that calls the binary file onbar_d
- Storage spaces (dbspaces, blobspaces) and logical logs to be backed-up or restored
- The ON-Bar catalog tables stored in the sysutils database
- IDS
- The XBSA interface shared library for each storage manager that your system uses
- Backup data on storage media
- The ON-Bar activity log
- The ON-Bar emergency boot file

Figure 1-9 shows the ON-Bar system and its components.

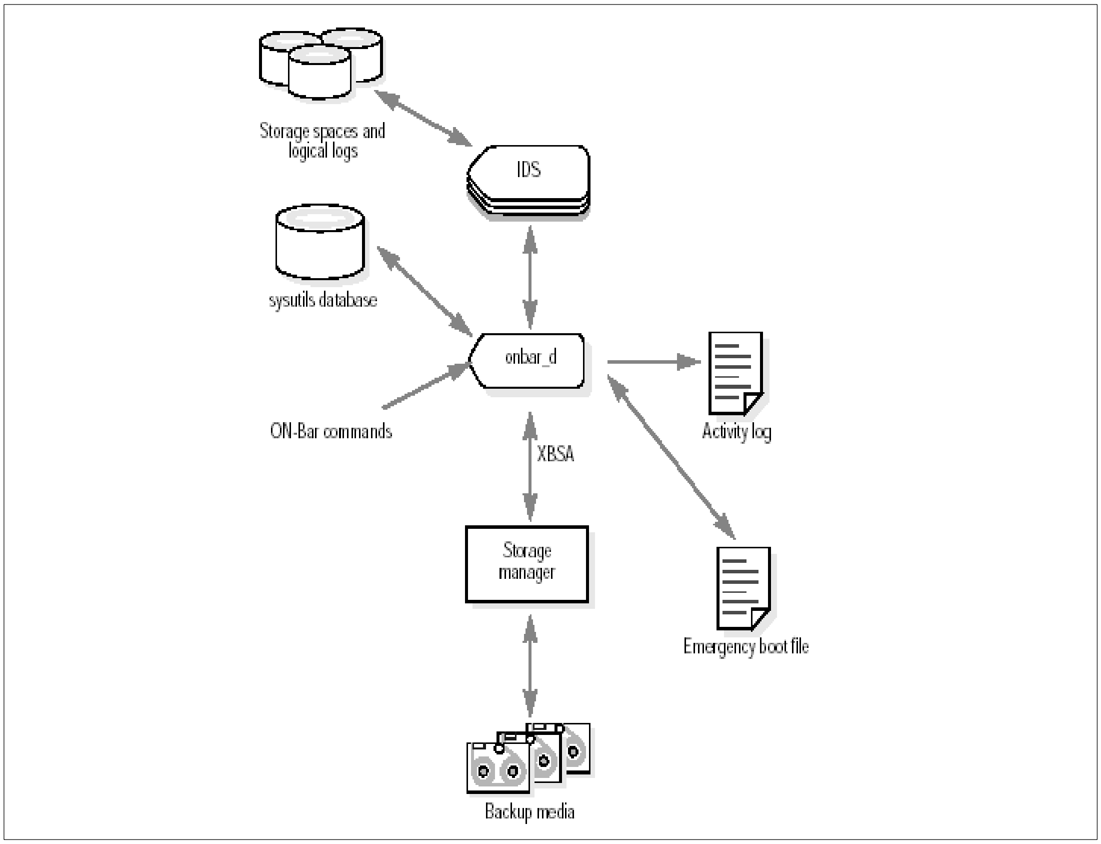


Figure 1-9 ON-Bar components

For speed and efficiency, ON-Bar can perform parallel backup and restore. For example, ON-Bar can back up multiple storage spaces concurrently. However, you can also configure ON-Bar to perform this task serially.

ON-Bar performs parallel backup and restore based on a configuration parameter, BAR_MAX_BACKUP. When ON-Bar receives a request, it determines how many objects are involved. If the request involves more than one object, ON-Bar creates a new onbar_d process for each object up to the limit that you specified in the BAR_MAX_BACKUP configuration parameter. Each new instance of ON-Bar creates a new XBSA session.

For more information about ON-Bar and Parallel Backup and Restore, see the *IBM Informix Backup and Restore Guide, Version 9.4*, G251-1240-00.

High-Performance Loader (HPL)

The HPL is a feature of the database server that allows you to load and unload large quantities of data efficiently to or from a database. The HPL lets you exchange data with tapes, data files, and programs, and converts data from these sources into a format compatible with Informix databases. The HPL also allows you to manipulate and filter the data as you perform load and unload operations. HPL is much faster than other load/unload utilities, such as SQL load/unload commands, onload/onunload, and dbload.

HPL uses a client-server architecture, and is composed of these three utilities:

- **ipload utility:** This is the graphical user interface, where you create load and unload jobs. These jobs are stored in the onpload database.
- **onpload utility:** The onpload utility performs the actual activity of converting and moving data. The onpload utility uses information from the onpload database to run the load or unload, and to convert the data.
- **onpload database:** The onpload database contains information that the onpload utility requires to perform data loads and unloads.

For more information about HPL, see the *IBM Informix High-Performance Loader User's Guide, Version 9.4*, G251-1255-00.

Extended features of IDS 9.x

All the features previously mentioned are common to both IDS Versions 7.x and 9.x. However, the extended features presented below are only valid for Version 9.x.

Before describing the extended features of IDS 9.x, we first should mention that IDS 9.x is not a Relational DataBase Management System (RDBMS) such as IDS 7.x. Rather, IDS 9.x uses an object-relational model and is referred to as an Object-Relational DataBase Management System (ORDBMS).

Relational Database Management Systems (RDBMS) support is limited to simple data types and pre-existing functionality defined in the database server. Data types provided by a traditional RDBMS are adequate to describe most business models. However, there are more complex models that work with more complex data, such as large arrays of data, points in a grid, maps, and images. Object Relational Database Managements Systems (ORDBMS) provide support for these complex data types. The functionality provided by a robust RDBMS, such as scalability, security, transaction recovery, and online backup and restore, is also required and provided in a ORDBMS. The object-relational model brings us a step closer to the business model. A ORDBMS takes advantage of the object-oriented concepts and addresses the shortcomings of the relational model.

IDS 9.x is a ORDBMS and provides both relational and object-oriented capabilities. The database server enables you to extend your database by defining new data types and user-defined routines (UDRs) to perform operations on those new data types. You can create UDRs in stored procedure language (SPL), C, or Java to extend the functionality of the database.

DataBlade modules are a packaging of data-type definitions and the functions that operate on them. IBM Informix and third-party vendors package some data types and their access methods into DataBlade modules or shared-class libraries. DataBlade modules allow you to store and manipulate complex data without having to create all the definitions and routines. They often plug directly into the components of the Dynamic Server for immediate use.

In Figure 1-10 we present the architecture of the IDS implementation of a ORDBMS. It is followed by further explanations of some of the extended features of IDS 9.x.

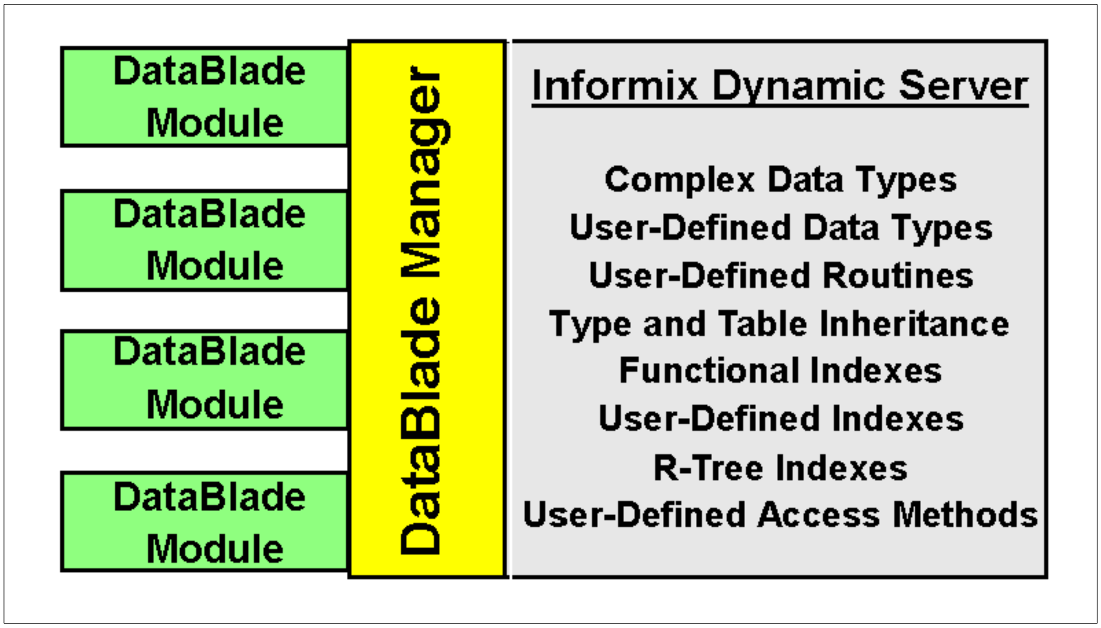


Figure 1-10 IDS 9.x architecture

User-Defined Routines (UDRs)

A routine is a collection of program statements that perform a specific task, and are used to extend the database capabilities. A user-defined routine is a routine that can be invoked in an SQL statement (such as a SELECT), by the database server or by another UDR. UDRs are also supported on IDS 7.3, but they can only be written using the Stored Procedure Language (SPL). The difference with IDS 9.x is that the UDRs can also be written in the C and Java programming languages.

Notes:

- To write routines in C, you need a C compiler and the IBM Informix ESQL/C compiler.
- To write routines in Java, you must have J/Foundation. You also must install the Java Development Kit (JDK).
- You may use wrappers for some other languages, such as C++.

The advantage is the flexibility that you get when writing your UDRs. Using powerful Programming Languages like C and Java provide the necessary tools to create routines for more complex tasks.

Creating UDRs using the SPL language is basically the same on both IDS 7.x and IDS 9.x, but in order to create a UDR using C or Java, we need to follow these steps:

1. Write the UDR using one of the supported Programming Language
2. Compile the routine and store the code in a shared library
3. Register the function in the database server with the CREATE FUNCTION or CREATE PROCEDURE statement.

User Defined Routines created using a Programming Language other than SPL are called external UDRs.

The best way to help you understand how to create an external UDR is to show you a simple example. Refer to Example 1-1, which is a simple routine that tests for negative numbers and returns a positive 1, otherwise it returns a zero.

Example 1-1 Creating a UDR using the C programming language

```
First of all we need to write the UDR using the chosen Programming Language, which in this
case is the C Language.

#include "/home/IDS9.40/incl/public/mi.h"
mi_boolean  is_negative(int x)
{
        if  (x<0)
                return 1;
        else
                return 0;
}

Notice that we include the $INFORMIXDIR/incl/public/mi.h header file. The set of header,
data types and functions files in the public directory are called the Datablade API.
After writing our UDR using the C language we need to compile it and make it a shared
library object. On Linux we use the GNU compiler, gcc, and these are the steps to compile
the source code (myudrs.c) and link it as a shared library:

gcc  -I/home/IDS9.40/incl/public -I/home/IDS9.40/incl -I/home/IDS9.40/incl/esql -c -fPIC
myudrs.c
ld -shared -soname myudrs.so -o myudrs.so -lc ./myudrs.o

These commands create a shared library called myudrs.so. Now we need to register the UDR in
the database server using dbaccess:

create function is_negative(integer)
returning boolean;
external name '/home/eduardo/myudrs.so(is_negative)'
language C
end function

Finally we can run our UDR on dbaccess:

execute function is_negative(-5);

This is the output that we get:

(expression)
t
```

For more information about creating UDRs using C, see these documents:

- *IBM Informix DataBlade API Programmer's Guide, Version 9.4*, G251-1258-00
- *IBM Informix DataBlade API Function Reference, Version 9.4*, G251-1257-00

User-Defined Data Types (UDTs) and Complex Data Types

On IDS 7.x the only data types available for use are the built-in data types. Those include data types such as CHAR, VARCHAR, INTEGER, and DECIMAL. On IDS 9.x we have all the built-in data types plus a new set of Extended data types. Since we can create our own data type, the maximum number of data types is unlimited.

The diagram in Figure 1-11 shows the set of data types available on IDs 9.x.

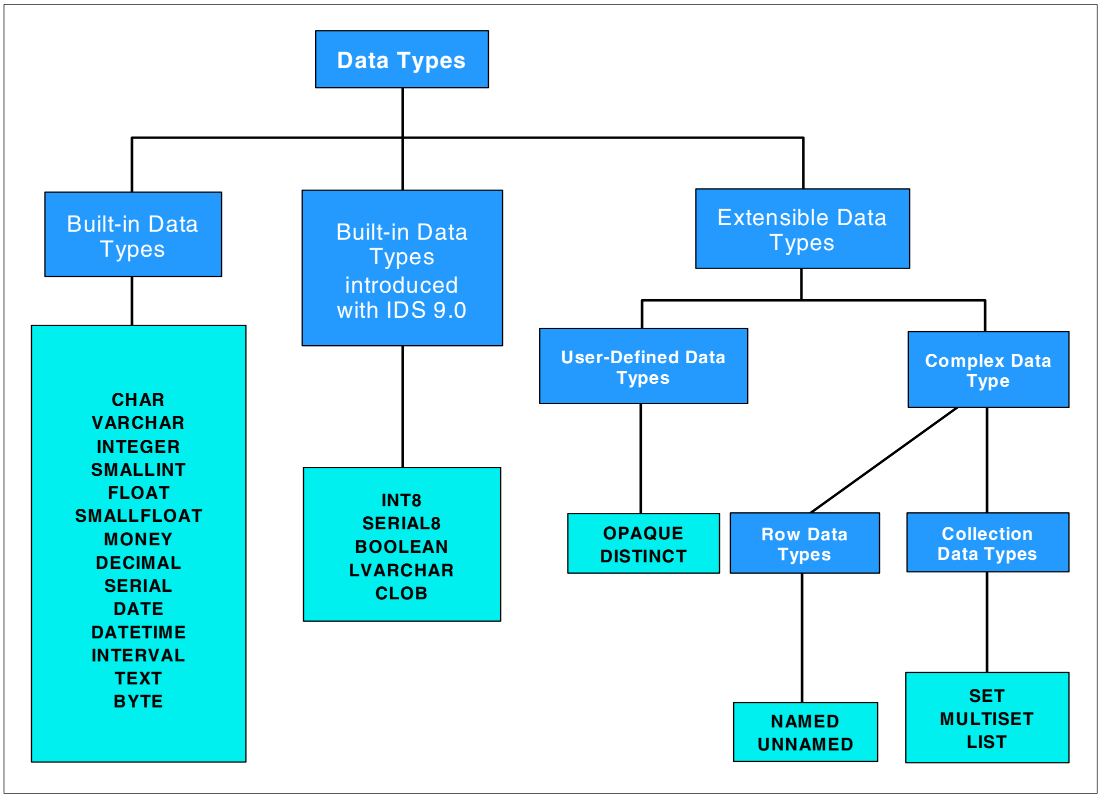


Figure 1-11 Data Types on IDS 9.x

Built-in data types usage is the same as on IDs 7.x, so here we focus on the description of the Extended data types:

- **Collection data types:** A Collection data type is composed of the SET, MULTISET, and LIST data types. They differ among each other based on the fact that the data stored is ordered/unordered and whether duplicates are allowed.
- **Row data types:** A Row data type can be created as NAMED or UNNAMED Row type. The main difference is that the NAMED Row type is identified by its name, while the UNNAMED one is identified by its structure. Another major difference is that NAMED Row type has inheritance properties and is used to create typed tables and columns.
- **User-defined data types (UDTs):** This set of data types is composed of DISTINCT and OPAQUE data types. Distinct types have the internal structure of an existing data type. When you create a distinct type, you base it on any of the built-in or user defined data types in the database server. You can define distinct functions that make the distinct type act differently than the source data type. An opaque type has a structure that is unknown to the database server. You must define the internal structure, functions, and operations.

There are other extended data types, but an explanation of all of them is beyond the scope of this book. However, to aid you in understanding some of what we have already described, we present Example 1-2, showing how to use a NAMED row type. The example shows the creation of a named row type and demonstrates how we can insert and select data.

Example 1-2 Creating and manipulating a NAMED row type

```
First we create the named row type:

CREATE ROW TYPE dimension_t(
length DECIMAL,
width DECIMAL,
height DECIMAL,
weight DECIMAL);

Notice that, in order to create the Row type, we use the CREATE ROW TYPE command.
Now, we can create a table using the named row type as the data type of the columns:

CREATE TABLE part (
part_id SERIAL PRIMARY KEY,
part_name VARCHAR(30),
part_desc LVARCHAR,
cost DECIMAL,
part_dimension dimension_t);

Here we insert one row into the table created:

INSERT INTO part VALUES(
0,
"J/M1862-old",
"Muffler, VW Bug",
168.34,
ROW(16.0,8.0,6.0,175.0)::dimension_t );

To query this row we run the following SELECT statement:

SELECT part_dimension.weight FROM part;
```

The NAMED Row type and typed tables (tables created using the named row types) also provide another particular feature of IDs 9.x: Table Hierarchy and Inheritance. Creating a hierarchy enables a database object to inherit the structure and behavior of an existing database object. The basic steps to achieve that is to first create the named row type hierarchy and then the table hierarchy.

In Example 1-3 we show how to create the named row type hierarchy and then the table hierarchy.

Example 1-3 Naming example

```
First we define the NAMED Row Type Hierarchy:

CREATE ROW TYPE person_t (
person_id SERIAL,
name name_t,
address address_t);

CREATE ROW TYPE employee_t (
salary MONEY(9,2),
hire_date DATE)
UNDER person_t;

Now, we create the table Hierarchy:

CREATE TABLE person
OF TYPE person_t;
```

```
CREATE TABLE employee
OF TYPE employee_t
UNDER person;

The subtable (employee) inherit all properties of the supertable (person) such as columns
and constraints definitions, storage options, triggers, indexes, and access methods.
```

The introduction of Extended data types provides a very flexible environment for the users. However, every function to manipulate (for instance, conversion to other data types) the user-created data types have to be coded through UDRs.

SQL Statement Cache

In earlier versions of IDS (such as IDS 7.x), each session generated and stored its own SQL statement information. Starting with Version 9.2, the database server allows sessions to share the SQL statement information that is generated. This allows multiple sessions that are executing identical statements to share the information stored in internal data structures. The SQL statement cache feature reduces memory utilization at the session level and eliminates the need to reparse and reoptimize statements that have been executed by other sessions.

The database server maintains the SQL statement cache in the virtual portion of shared memory. As SQL statements are executed by various sessions, they are parsed, optimized, and stored in this cache. Only data manipulation language (DML) statements (SELECT, INSERT, UPDATE, and DELETE) are cached. The main purpose of the SQL Statement Cache is to reduce the usage of memory, but some performance gains might also be noticed.

Figure 1-12 shows a comparison between sessions utilizing their private memory area and sessions using a shared SQL Statement memory area.

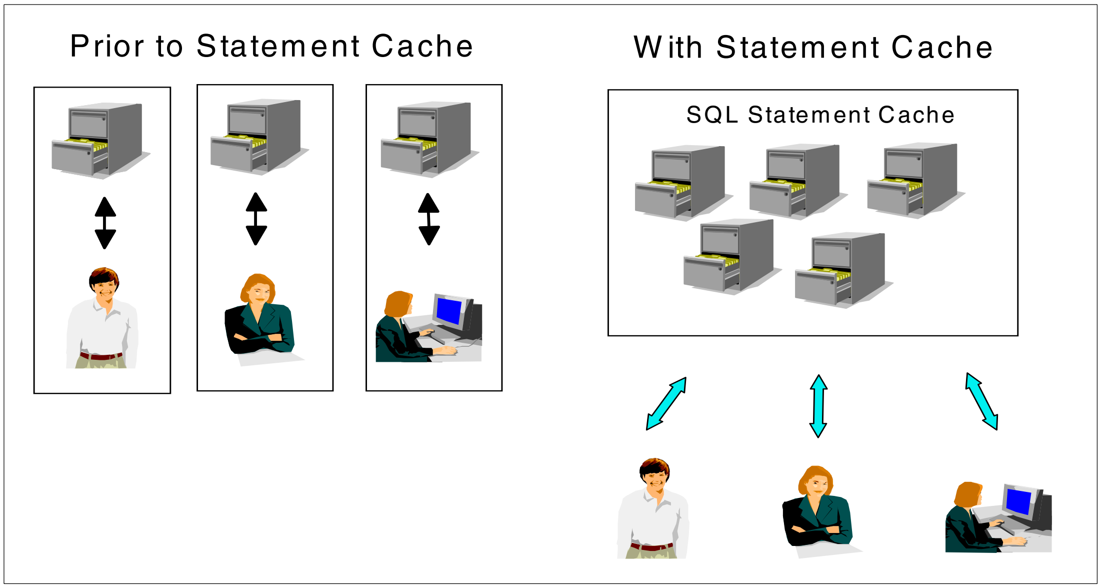


Figure 1-12 Memory usage comparison

We can configure SQL Statement cache at the global level and at the application level. To configure it globally, there are two configuration parameters (onconfig) that need to be specified:

- **STMT_CACHE:** A STMT_CACHE value of 0 disables statement caching. A value of 1 enables the statement caching but is, by default, turned off for sessions. A value of 2 enables statement caching and, by default, turns on caching for sessions. The default value for this parameter is 0.
- **STMT_CACHE_SIZE:** Set the STMT_CACHE_SIZE configuration parameter to the size, in kilobytes, to use for the statement cache. The default is 72 kilobytes. Regardless of the parameter value, if the cache becomes full and all the statements are active, new statements cause the cache to grow! When an SQL statement is no longer needed in the cache, the memory is freed until the size of the statement cache is less than or equal to the STMT_CACHE_SIZE.

You can set the SQL Statement cache at the application or session level, using either the STMT_CACHE environment variable or the SQL Command SET STMT_CACHE. However it only takes effect if the global configuration parameter STMT_CACHE is set to a value greater than 0.

Fuzzy checkpoints

The checkpoint is a major event on the database server. It can be defined as a synchronization between memory and disk. Pages in memory are written on disk and during the checkpoint operation the system is frozen and no database activity can occur. The flush of these pages is very costly and depending on the number of pages in memory the system can be unavailable for a critical period of time.

To reduce the checkpoint duration IBM introduced on IDS 9.2 a new feature called Fuzzy Checkpoints. The old checkpoint method still exists, but now it is referred as *full* or *sync* checkpoint. The idea to solve this performance issue is to write fewer pages to disk during a checkpoint. To accomplish this, a certain subset of database operations have been designated as fuzzy. The buffers modified by these fuzzy operations are flagged, and are not written to disk when a checkpoint occurs. Fuzzy operations are inserts, updates and deletes. However, they are only flagged as fuzzy operations if they are executed against a database that has logging and the buffered page to be changed is not considered old (as determined by an internal timestamp comparison).

The problem with this approach is that some pages in memory are not going to be consistent with the disk. Every change that is made to these pages is tracked in the logical logs, so that in case of a system crash the server will take more time processing these pages and they can get logically consistent again.

Note: All manuals mentioned in the foregoing sections can be found at:

http://www-3.ibm.com/software/data/informix/pubs/library/ids_94.html

2

Installing and configuring IDS V9.40

In the previous chapter we described some of the functions and features of IDS that make it a very powerful DBMS. However, to get maximum benefit from IDS requires that you install and configure it properly. In this chapter we provide some guidelines that will enable you to do this quickly and easily.

To develop the installation guidelines, the redbook team installed and configured IDS and then documented the process. We provide step-by-step instructions along with screen captures taken from our installation to make it even easier for you to follow.

Topics included in this chapter are:

- The new features of IDS 9.40 and how you can benefit from them
- Installation of IDS and configuration of a server instance
- Configuration of the IDS environment for integration with WebSphere

2.1 Taking advantage of new IDS 9.40 features

IDS 9.40 is the latest database server release available. It is a state-of-the-art ORDBMS that provides the necessary components for enterprise and workgroup computing needs. It contains the robust RDBMS functionality and performance of IDS 7.x, and all the extended features and object-oriented architecture of previous IDS 9.x releases. In addition, IDS 9.40 also introduces new features that significantly improve performance, reliability, and scalability, and make IDS 9.40 the most suitable release for today's highly available and demanding global e-business environments.

In this section we describe features introduced in IDS 9.40, focusing in the following major areas of enhancement:

- Security
- Increased scalability and usability
- Performance
- Data availability
- SQL and extensibility
- Backup and restore

Security enhancements

In today's extremely competitive market, security is an important consideration for every enterprise. System administrators are increasingly concerned about the security of the data within their corporate databases. This concern is particularly directed at that portion of the overall data management system that is most likely to be a target of interest by unauthorized users: the network.

IDS 9.40 supports encrypting data transmissions over the network using the encryption communication support module (ENCCSM). This module provides complete data encryption with the openSSL library, and has many configurable options. A message authentication code (MAC) is transmitted as part of the encrypted data transmission to ensure data integrity. A MAC is an encrypted message digest. The encryption algorithms use openSSL 0.9.6 as the code base. Distributed queries can also be encrypted.

The security enhancements provide secure client connections to the server, secure server to server connections, ability to prohibit unencrypted connections, and support for plug-in authentication modes.

Increased scalability and usability enhancements

IDS 9.40 features some new scalability and usability enhancements to meet the unpredictable challenges of today's on demand e-business environments. These features provide a much more scalable and easy-to-use database server for systems administrators. In the following sections we provide examples of some of the new enhancements.

Increased maximum size of chunks and maximum number of chunks

Enterprise systems are using databases to store more and more complex files such as documents, graphics, spreadsheets, HTML pages, and video. As a result, database tables are steadily increasing in size. Corporate consolidation, globalization trends, and management requirements for keeping data online for a longer periods of time, are also contributing to the continuous growth in the size of databases.

Pre-9.40 versions of IDS have a instance size limitation that is increasingly impacting large database installations. This limitation is associated with two factors, the maximum size of a chunk (2GB) and the maximum number of chunks (2047). These result limit the maximum size of an instance to around 4 TB (terabytes).

There are two categories of problems associated with the pre-9.40 IDS chunk size and chunk number system limits. These are the maximum instance size and the associated system administration effort required to deal with these two limits.

For IDS 9.40, these limits have been increased. The maximum chunk size is now 4 TB, and the maximum number of chunks is 32766. With these increases, the maximum size of an instance has been increased to 128 PB (petabytes). This equates to 128 quadrillion bytes!

As disk drive capacity moves into the terabyte range, the entire drive can be allocated as a single chunk to contain this new 4 TB chunk size. And now, up to 32,767 of these mega-chunks (4 TBs each) can be handled by a single instance of IDS 9.40.

Increased number of dbserver aliases

IDS was originally designed to provide only one server alias for each protocol. However, customers have been using server aliases for other purposes. For example, many customers use different server aliases for the different applications that connect to the server. And that has created a requirement to support the configuration of multiple server aliases. To meet this requirement, the maximum number of DBSERVERALIASES has been increased from 10 to 32. Since a single line in the onconfig is limited to 255 characters, 32 aliases could overflow this limit. To prevent that from happening, multiple lines for DBSERVERALIASES can now be used in the onconfig.

Performance enhancements

Performance is always the major topic of discussion when a product upgrade is considered for an enterprise system. With the extensibility features in versions 9.x, the database server was also generally much larger than IDS 7.x. This became a concern for many system administrators because of the potential impact on performance. Even though their new business models required more and more of the features provided only on a ORDBMS, this became and obstacle and some were hesitant to migrate to the new release.

However, with the performance enhancements introduced on IDS 9.40 that obstacle is gone. Performance on IDS 9.40 is expected to be the same or better than IDS 7.x, even with the new functionality. Internal benchmarks have validated this performance boost. Now administrators can be comfortable with the migration to IDS 9.40 and their ability to take advantage of the new functions and features.

To meet the performance requirements, a number of fundamental database server resource management algorithms have been substantially reworked and streamlined. The following sections describe some of the primary IDS components and the changes that were made to help achieve the performance goals.

Optimized index cleaning

The database server uses an index to quickly find a particular row of data. However, these indexes need to be maintained and kept clean to perform at maximum levels. What does it mean to be kept clean?

An index is arranged as a hierarchy of pages in a data structure called a B-tree and is composed by a set of nodes that contain keys and pointers which are arranged in a balanced tree structure hierarchy. The leaves of the tree are linked sequentially to also enable fast sequential access to data.

The B-tree is organized into levels. The leaf nodes contain pointers, or addresses, to the actual data. The other levels contain pointers to nodes on lower levels that contain keys which are less than or equal in value to the key in the higher level. Figure 2-1 illustrates the structure of a B-tree.

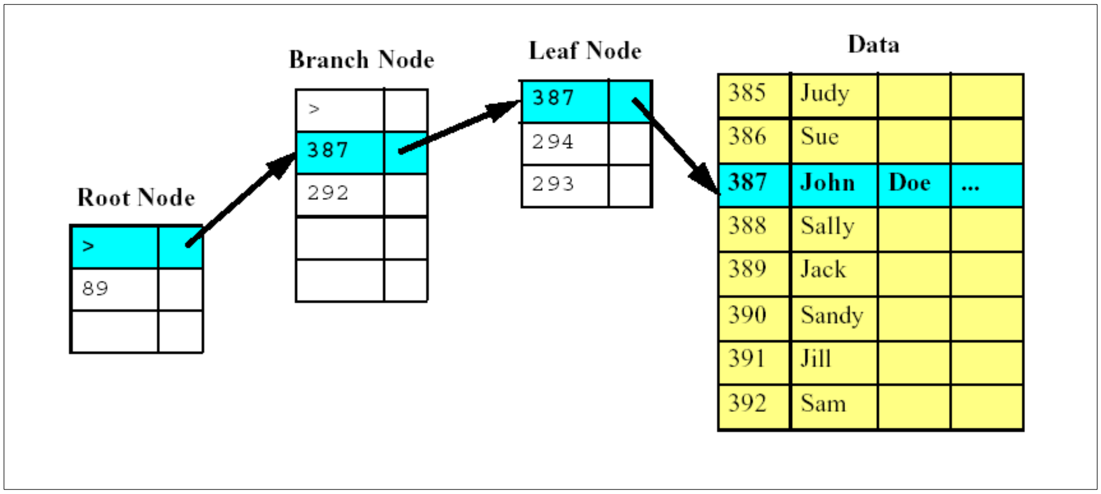


Figure 2-1 B-tree structure

When a DELETE operation is performed by a user, the B-tree entries also have to be removed and the B-tree itself needs to be rebalanced. This has been the responsibility of a system thread called *btcleaner*. Basically when a deleted item is committed, its page number is put in the B-tree cleaner queue. The btcleaner thread reads the pool occasionally and removes all committed deleted items on each page.

However, there were some potential problems with the B-tree cleaner system. Here are some examples:

- There has always been only one B-tree cleaner thread and during heavy OLTP processing, which could get overwhelmed with cleaning requests.
- Pages that were freed and then reused, often contained left-over data that could result in invalid requests for btcleaner.
- A single B-tree cleaner queue often resulted in contention.
- Priority issues needed to be considered with submitted requests.
- Long lists of committed deleted items left a bloated index, which reduced available space and, more importantly, introduced performance problems.

Solving these issues by changing the B-tree cleaner system was a complex task, which would require many parts of the source code to be rewritten. To avoid this issue, IDS 9.40 introduced a new index cleaning method, and eliminated the btcleaner thread. The new B-tree cleaning system is called the *B-tree scanner*.

The B-tree scanner uses new and different methods to solve the issues with the B-tree cleaner. For example, it improves transaction processing for logged databases when rows are deleted from a table with indexes, and it removes deleted index entries and rebalances the index nodes. Here is how it accomplishes that:

- The workload for cleaning indexes is now prioritized. For example, the index that causes the server to do the most work will be the next index cleaned.
- An index will have its leaf level examined, searching for deleted items. The B-tree scanner will test lock the found item, then do a foreground remove of the item and possibly compress the page.
- Dynamic configuration of threads to allow for configurable workloads. New B-tree scanner threads can be added to, or removed from, the database server. The new threads can also be tuned through new *onmode* options.

Some new terms have also been introduced, with which administrators should become familiar:

- **Dirty count or hits:** The number of times a user or administrative thread has encountered an committed deleted item while performing work.
- **Hot list:** The list of indexes that need to be cleaned
- **B-tree scanner:** The new threads that are responsible to clean the indexes.

Improved priority management for the buffer manager

Memory management and tuning is a very important requirement to achieve high performance when a large number of users are processing database activities on the same database server. IDS uses *Least Recently Used Queues* (LRU queues) to manage access to the buffer pool in the resident portion of shared memory. The free or unmodified page list is referred to as the FLRU queue of the queue pair, and the modified page list is referred to as the MLRU queue. The two separate lists eliminate the need to search a queue for a free or unmodified page.

When a user thread needs a buffer, the server selects one from the Least Recently Used end of a FLRU queue. If the page is subsequently modified, it is placed in a buffer at the Most Recently Used end of the MLRU queue. If the page has been read but not modified, the pointer to the buffer is returned to the Most Recently Used end of the FLRU queue.

The LRU queues are divided into different priority regions. Prior to IDS 9.40, these were: HIGH, MED_HIGH, MED_LOW and LOW. With this prior buffer management system, some design assumptions were made to categorize a page in one of the priority levels. For instance, root nodes of an index were always set as priority HIGH, branch nodes were given MED_HIGH priority, and leaf nodes and data paged were set a MED_LOW priority.

This categorization resulted in many OLTP systems using multiple indexes per table, and could result in the index using more space than the actual data. With the algorithm used to set the LRU priority, the buffer pool could become swamped with index pages. Then the data pages would be displaced to disk, which typically results in degraded performance.

Figure 2-2 illustrates this scenario.

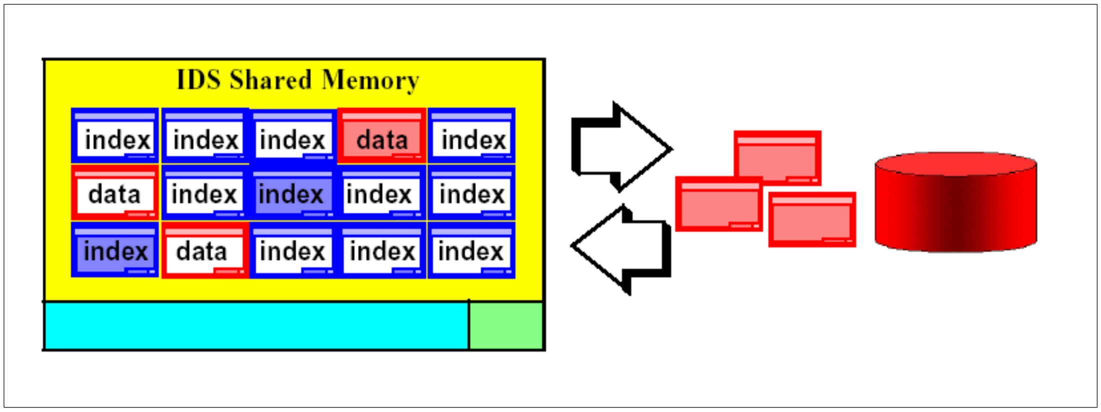


Figure 2-2 Buffer pool swamped with index pages

With IDS 9.40, the prioritization of pages has been changed, and is based on a simple but efficient approach. Pages that are more frequently utilized by users have higher priorities, and would be candidates to stay in the buffer pool for a longer period of time.

Now the four priorities have been reduced to two:

- **HIGH:** Managed by FIFO1 queue
- **LOW:** Managed by LRU queues

The maximum size of HIGH priority band is fixed, whereas with prior versions it could grow indefinitely. And now, the HIGH priority buffers can no longer consume the entire buffer pool.

More precise LRU maximum and minimum settings

The configuration parameters LRU_MIN_DIRTY and LRU_MAX_DIRTY determine how frequently the MLRU queue buffers are flushed to disk. Depending on how these parameters are set, most of the synchronization between memory and disk is performed during the checkpoint and can cause the system to encounter a significant slow down.

The LRU_MAX_DIRTY parameter defines the maximum percent of an LRU queue pair that is permitted to be dirty before page-cleaner activity begins. The LRU_MIN_DIRTY defines the minimum percent of an LRU queue pair that is permitted to remain modified after the page cleaning activity finishes. So, if these parameters are set to a small value more page flushes will occur between the checkpoints, leaving less I/O work to be performed by the checkpoint itself.

The default values for the LRU_MAX_DIRTY and LRU_MIN_DIRTY parameters are 60 and 50 respectively. In previous IDS versions they only accepted integer values, which could result in performance problems when there was a huge buffer pool. This is because 1% (the minimum integer value) of a large buffer pool could represent a large value that could impact the checkpoint duration.

In IDS 9.40 these two parameters can be configured with decimal values. Now you have more control and precision in tuning the frequency of the disk flush, and consequently the duration of the checkpoints.

Data availability enhancements

As we mentioned in the previous chapter, HDR and ER play a very important role in supporting high availability of data. In IDS 9.40 new features and extensibility are introduced for both ER and HDR.

First let us look at the new data types and objects supported for enterprise replication. ER now supports the following data types: ROW, NAMED, and UNNAMED, and collection data types (LIST, SET, and MULTISET).

HDR now supports replication of the following extended objects: All built-in and extended data types, user-defined routines (with some limitations), user-defined types, and R-tree indexes (a different access method).

With IDS 9.40, ER and HDR can co-exist on the same database server, which was not possible on prior releases. Therefore, if high availability is critical, you can use HDR in conjunction with ER. HDR uses synchronous data replication between two database servers. For example, a primary serve can participate in enterprise replication, and a secondary server can be read-only and not participate in enterprise replication. If a primary server in an HDR pair fails, you switch the secondary server to be the primary server, allowing it to participate in enterprise replication. Client connections to the original primary server can be automatically switched to the new primary server

Figure 2-3 shows a hypothetical scenario of HDR and ER configured to work together.

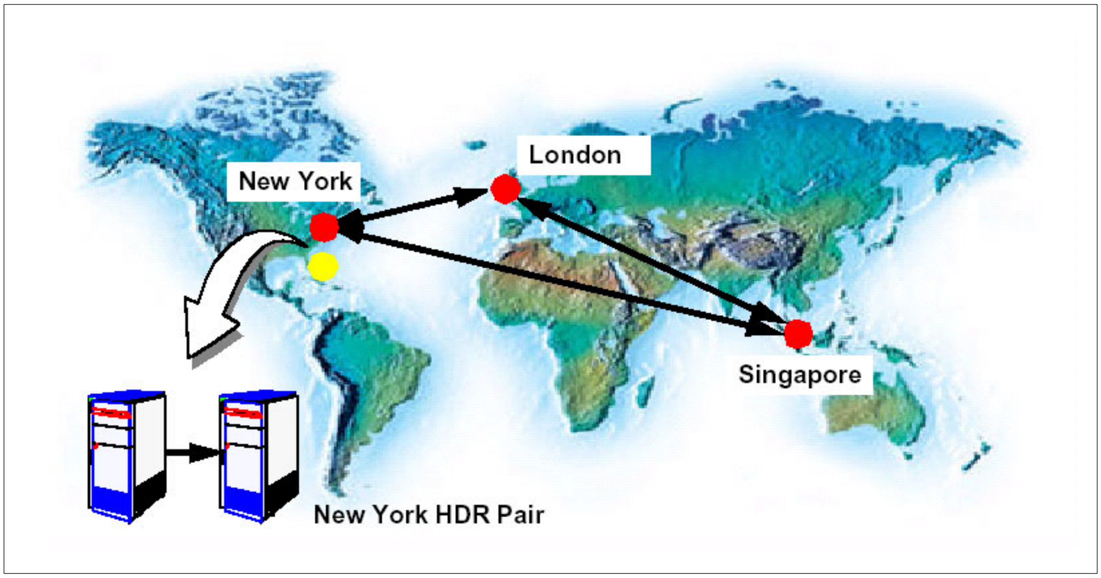


Figure 2-3 HDR and ER coexistence scenario

SQL and extensibility enhancements

New SQL features are also introduced on IDS 9.40. These enable support for more ANSI standard features, improve compatibility for migrating from other DBMS vendors, and provide better integration with other IBM products such as WebSphere. In this section we describe some of the primary enhancements.

Triggers on views

Applications, for security and other reasons, often use views to hide portions of a table from users. Views are a good solution for applications that need to service a large set of users but without exposing all available data. However, prior to IDS 9.40, views were non-updatable so users could only query the data. Thus, other applications were required for this purpose.

With IDS 9.40 a view can now be updated. This is accomplished with the use of new INSTEAD OF Triggers on views. With the INSTEAD OF trigger feature one can perform DML operations [insert, update or delete] on a non-updatable [complex] view, and thus direct the server to perform operation on the underlying tables of the view. Example 2-1 shows an insert on a view.

Example 2-1 Inserting data on a view using INSTEAD OF trigger

```
CREATE TABLE dept(
deptno INTEGER PRIMARY KEY,
deptname CHAR(20),
manager_num INT);

CREATE TABLE emp(
empno INTEGER PRIMARY KEY,
empname CHAR(20),
deptno INTEGER REFERENCES dept (deptno),
startdate DATE);

/* The view manager_info lists all the managers for
each department */

CREATE VIEW manager_info AS
SELECT d.deptno, d.deptname, e.empno, e.empname
```

```
FROM emp e, dept d
WHERE e.empno = d.manager_num;

/* The INSTEAD OF trigger manager_info_insert handles inserts on
the manager_info view */

CREATE PROCEDURE instab(dno INT, eno INT)
INSERT INTO dept (deptno, manager_num) values(dno,eno);
INSERT INTO emp(empno, deptno) VALUES(eno, dno);
END PROCEDURE;

CREATE TRIGGER manager_info_insert
INSTEAD OF INSERT ON manager_info REFERENCING NEW as n
FOR EACH ROW
(EXECUTE PROCEDURE instab(n.deptno, n.empno));

INSERT INTO manager_info(deptno,empno) VALUES(08,4232);

/* The insert on the view should work fine at this point */
```

Enhanced SELECT statement syntax

Existing applications rely on the ability to order query results based on values that are not in the select list, or the results need to be ordered by an expression. This was not permitted on prior versions of IDS. Now, on IDS 9.40, the ORDER BY clause is more flexible — which now makes this query valid, as shown in Example 2-2.

Example 2-2 SELECT statement with enhanced ORDER BY clause

```
SELECT d.dept_num
FROM dept d, employees e
WHERE d.dept_num = e.dept_num
GROUP BY d.dept_num
ORDER BY AVG(e.salary);
```

ANSI join syntax

The syntax of the SELECT statement has been enhanced to support the ANSI/ISO syntax for cross joins, right outer joins, and full outer joins. The keywords CROSS, RIGHT, and FULL, are now supported in the context of queries that join two or more tables and can be used with the ON clause. This makes the new syntax more compliant with the SQL-99 standard, and also enables enhanced integration of IDS with WebSphere.

Stored procedures return parameters

Prior to IDS 9.40, the return values of a stored procedure did not have a label for the column name. Now you can specify names for each one of the parameters that are returned. See Example 2-3 for a comparison.

Example 2-3 Code samples with and without named return parameters

```
/* Code Sample without Named Return Parameters */

create table t1 (c1 int , c2 int);
insert into t1 values (10,20);

create procedure proc()
returning int, int;
define n ,m int ;
FOREACH curA FOR SELECT * INTO n,m FROM t1
return n,m with resume;
```

```
END FOREACH;
end procedure;

execute procedure proc();
 => Dbaccess Output:
-------------------------
(expression) (expression)
          10           20

/* Code Sample with Named Returned Parameters */

create table t1 (c1 int , c2 int);
insert into t1 values (10,20);

create procedure proc()
returning int as procc1, int as procc2;
define n ,m int ;
FOREACH curA FOR SELECT * INTO n,m FROM t1
return n,m with resume;
END FOREACH;
end procedure;

execute procedure proc()
 => Dbaccess Output:
--------------------------
procc1          procc2
    10              20
```

Sequence objects

IDS 9.40 introduced new DML statements (CREATE SEQUENCE, ALTER SEQUENCE, RENAME SEQUENCE, DROP SEQUENCE) for sequence generators. A sequence is a user defined database object that generates a sequence of numbers in a monotonically ascendent/descendent order based on the options specified when the sequence was created. Concurrent users can generate sequence numbers (unique, if required) using a sequence generator. A sequence object provides functionality that is similar to the SERIAL type in IDS. Example 2-4 shows the syntax of the new DML statements.

Example 2-4   SQL syntax for SEQUENCE objects

```
CREATE SEQUENCE <sequence_name> [INCREMENT [BY] <increment_value>]
[START [WITH] <start_value>]
[MAXVALUE <max_value> | NOMAXVALUE]
[MINVALUE <min_value> | NOMINVALUE]
[CYCLE | NOCYCLE]
[CACHE <cache_num> | NOCACHE]
[ORDER | NOORDER]

ALTER SEQUENCE <sequence_name>
[INCREMENT [BY] <increment_value>]
[MAXVALUE <max_value> | NOMAXVALUE]
[MINVALUE <min_value>| NOMINVALUE]
[RESTART [WITH] <restart_value>]
[CYCLE | NOCYCLE]
[CACHE <cache_num> | NOCACHE]
[ORDER | NOORDER]

DROP SEQUENCE <sequence_name>

RENAME SEQUENCE <sequence_name> TO <new_sequence_name>
```

Backup and restore enhancements

The backup and restore mechanisms on IDS 9.40 have also been enhanced to better facilitate many of the administration tasks.

Full use of storage media

Prior to version 9.40, the system administrator was required to specify the size of the storage media that was being used by backup and restore utilities, and risked wasting storage space when the size of the media was not correctly specified. The size of the storage media now is specified through configuration parameters or command line options. The backup and restore utilities include: ontape, onload/onunload, dbimport/dbexport, and HPL. Except for HPL, you can now specify a value of 0 for the size of the storage media. This means that the storage media is used to the end by all server utilities mentioned above.

Increased file size limit

Prior to version 9.40, there was a file size limit of 2 GB. This limitation applied to many server utilities, such as the UNLOAD and LOAD statements of SQL, onspaces, ontape, and dbimport and dbexport utilities. This forced systems administrators to partition data across multiple files. The new file size limit is 4 TB, and applies to all utilities mentioned above.

Chunks redirection during a cold restore

Prior to version 9.40, a backup of a server could only be restored to another server having identically named devices or files as the original server. This was a very limiting restriction considering that sometimes the device might not be available. This was also an impact anytime the system administrator needed to restore a server instance to a different server.

IDS 9.40 eliminated that issue by enabling the renaming of chunks during a cold restore. A cold restore is a restore that occurs when the root dbspace, or the dbspace that holds the logs, is inaccessible, and it is performed with the server in offline mode. During a cold restore you can now change the target chunks using either the ON-Bar or ontape utilities. The chunk remapping is done directly through the command line, or indirectly through a referenced file. Example 2-5 shows an example of command line chunk mapping.

Example 2-5 Command line chunk mapping

```
/* Command line chunk mapping */

Onbar
onbar -r -rename -p /dev/dsk/c1s0t2d1 -o 0
-n /ifmx/instance2/rootdbs.c0 -o 20000

Ontape
ontape -r -rename -p /dev/dsk/c1s0t2d1 -o 0
-n /ifmx/instance2/rootdbs.c0 -o 20000

/* The actual syntax is */

onbar
onbar -r [{-rename -p <old chunk path> -o ,old offset>
-n <new chunk path> -o <new offset>}...]

ontape
ontape -r [{-rename -p <old chunk path> -o ,old offset>
-n <new chunk path> -o <new offset>}...]
```

Example 2-6 shows an example of file-based chunk mapping.

Example 2-6 File-based chunk mapping

```
Onbar
onbar -r -rename -f chunks_file

Ontape
ontape -r -rename -f chunks_file

/* chunks_file: place each entry on a separate line
<old chunk path> <offset> <new chunk path> <offset> */

/dev/dsk/c1s0t2d1 0 /ifmx/inst2/rootdbs.c0 20000
/ifmx/inst1/blobspc.c1 10000 /ifmx/inst2/blobspc.c1 0
```

2.2 Installing IDS 9.40 on SuSE Linux V8

In this section we describe the procedures to install and configure IDS on Linux, along with some basic IDS administration commands that are used. We chose IDS 9.40 for all the extensibility benefits and new features previously discussed, and because it is the newest and best IDS version available. The actual package installed was IDS 9.40 with J/Foundation. This enabled us to run Java procedures on the database server. However, for simplicity, the term IDS 9.40 is used throughout this book. We chose Linux as the operating system because it is an open source platform that is powerful, reliable, and secure. We used SuSE Linux 8.0 for the installation of both IDS 9.40 and WebSphere Application Server 5.0.

We have divided our discussion of this topic into two main sections: **Planning** and **Installing**.

Planning

Before installing the products there are some pre-installation tasks that need to be performed and verified.

System requirements

Here are some hardware and software recommendations when installing IDS on Linux:

- Pentium® III
- CPU - Intel® 32 bits
- Minimum of 500 MB of disk space
- Minimum of 128 MB of memory
- Linux kernel 2.4.7 or higher

Kernel parameters

We recommend that some of the OS kernel parameters are tuned according to the machine notes. These are the values used to build IDS 9.40, and is a good starting point. Some applications may require these parameters to be tuned differently.

- SHMMAX: 33554432
- SHMMIN: 1
- SHMMNI: 128
- SHMSEG: 128
- SHMALL: 4194304
- SEMMNI: 128
- SEMMSL: 250
- SEMMNS: 32000
- SEMOPM: 32

These parameters, and other valuable information, are documented in the release notes directory, located in $INFORMIXDIR/release/en_us/0333. We recommend that you read these documents, especially the following files:

- **ids_unix_install_docnotes_9.40.html:** The documentation notes file for your version of this guide describes features that are not covered in the guide or that were modified since publication.
- **ids_unix_release_notes_9.40.html:** Release notes files describe compatibility issues, feature differences from earlier versions of IBM Informix products, and how these differences might affect current products. Release notes also contain information about any known problems and their workarounds.
- **ids_machine_notes_9.40.txt:** The machine notes file describes any special actions that you must take to configure and use IBM Informix products on your computer. Machine notes are named for the product described.

Creating the users and groups

In this section we discuss the creation of users and groups.

Creating the user and group Informix

If you are installing IDS for the first time on a determined host machine, you need to create a *user* called informix and a *group* called informix. Their IDs should be greater than 100.

On Linux we used the `useradd` and `groupadd` commands to accomplish that:

1. Log in as user **root** and start a terminal session.
2. Create the group informix using the groupadd command. For example:

   ```
   groupadd -g 200 informix
   ```

3. Create the user informix using useradd. For example:

   ```
   useradd -u 200 -g informix -d /home/informix informix
   ```

Note: The user informix is the database equivalent of the UNIX® or Linux root account, so that anyone logged in as user informix has complete access to any IBM Informix products and databases. Keep the user informix password confidential.

IBM Informix products use group informix internally to control database access. Make user informix the *only* member of group informix. Any person who belongs to group informix is a database server administrator.

Creating the user itso — used with our sample applications

Notice that creation of this user is not mandatory for the installation of IDS. We are creating the user *itso* because it will be used later as we develop the sample applications in this book.

On Linux we used the `useradd` to accomplish that:

1. Log in as user **root** and start a terminal session.
2. Create user itso using useradd. For example:

   ```
   useradd -g users -d /home/itso itso
   ```

Installation directory and environment variables

Now you must choose a directory where the product will be installed. After that, the INFORMIXDIR and PATH environment variables should be set. For instance, in our case the informix directory is **/home/IDS9.40**. We have to set this in our INFORMIXDIR and include it in the PATH variable as well. On our Linux installation we used the bash command shell and we set the variables as follows:

```
export INFORMIXDIR=/home/IDS9.40

export PATH=$INFORMIXDIR/bin:$PATH
```

If you use C shell, the commands to set the variables are:

```
setenv INFORMIXDIR /home/IDS9.40

setenv PATH ${INFORMIXDIR}/bin:${PATH}
```

Installing

The process of installing IDS 9.40 varies depending on the product distribution format that you have. Usually IBM Informix products are distributed in CD-ROM, where the UNIX bundle installer and other files are already expanded. However, if you download the product from an FTP site, you might get the product in a cpio, tar, or Redhat Package Manager (RPM) format.

In our case we downloaded the product from an internal site and it came in a tar format. The only difference is the command that you run to extract the bundle installer and installation files. Here we show how to expand these files using the tar command and then how to use the installation script to install IDS 9.40, using some screen captures to illustrate this process.

> **IMPORTANT:** Refer to Chapter 13, “Implementation hints and tips” on page 295 before you begin your IDS installation. For SuSE 8.0, a new library needs to be installed.

1. Since our product came in a tar file we needed a temporary space to expand the distribution files and the installation scripts. It requires about 500 MB of disk space to expand these files. Use the tar command to extract the files.

   First, log in as user root. Then copy the .tar file you downloaded into the temporary directory. Now go to the temporary directory and run the tar command that uncompresses the file, using the following commands:

   ```
   cd /home/informix/tmp

   tar -xvf IDS9.40.tar
   ```

   This following command can then be used to list all the files being extracted, and when it finishes the temporary directory will contain the following files and directories:

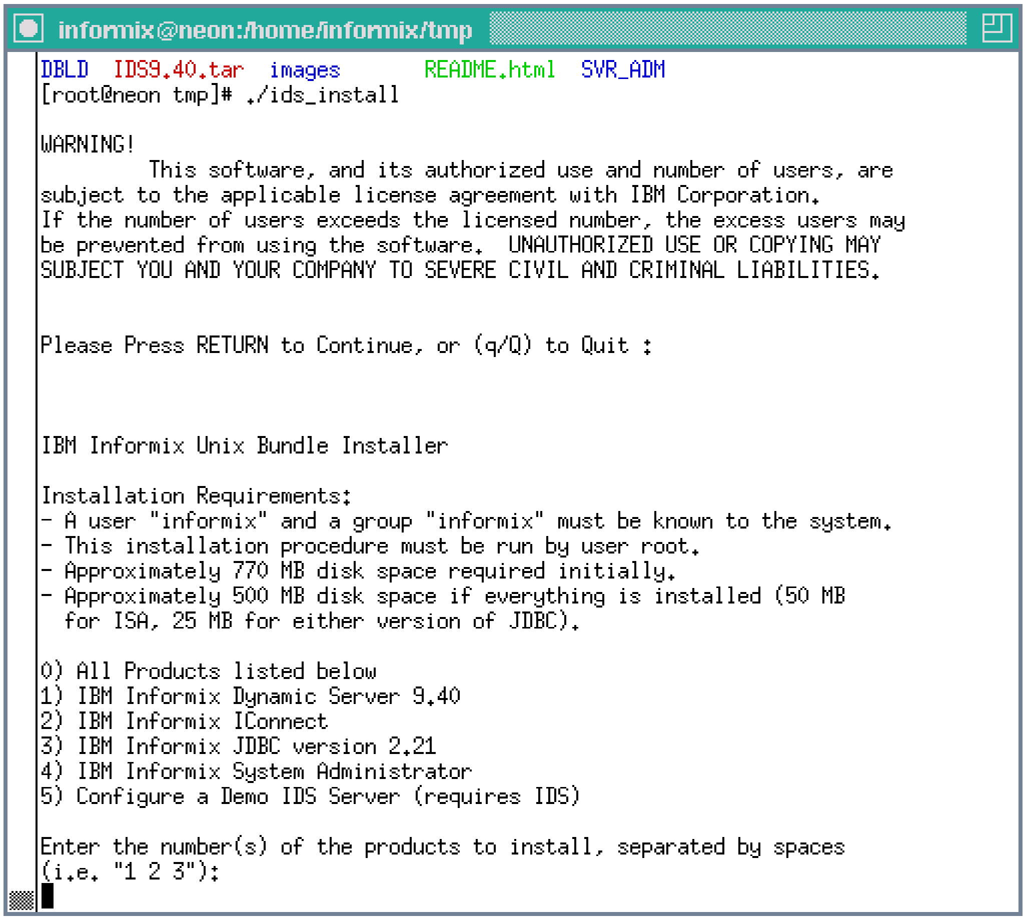
   ```
   [root@neon tmp]# ls
   BDMA  DBLD  ICONNECT ids_install  images  JDBC  README.html  SERVER
   SVR_ADM
   ```

   > **Note:** If your product distribution comes in a cpio format, run this command in the temporary directory:
   >
   > ```
   >    cpio -icvdumB < IDS9.40.cpio
   > ```
   >
   > For RPM distributions, run the command:
   >
   > ```
   >    rpm -iv IDS9.40.rpm
   > ```
   >
   > If you have IDS 9.40 on CD-ROM, you should already have this structure and won’t need the extra disk space.

2. Run the ids_install bundle installer by entering the following command:

   [root@neon tmp]# ./ids_install

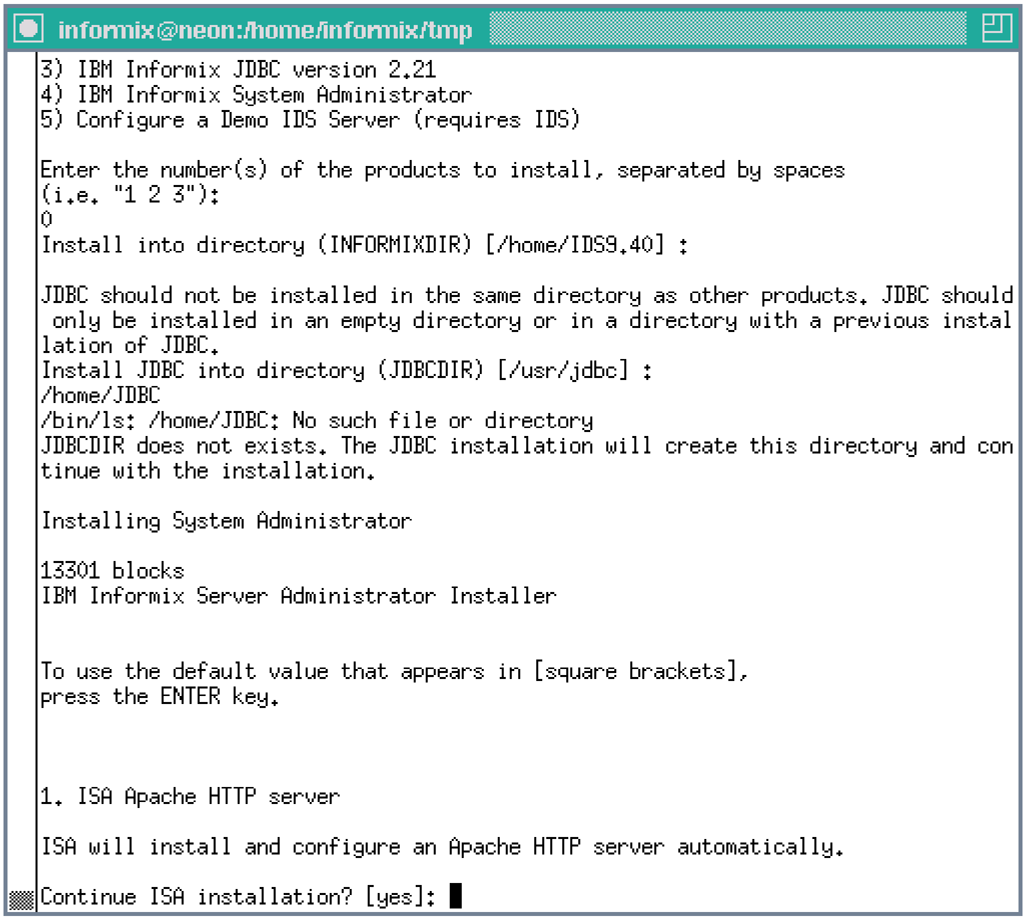
The installation menu shown in Figure 2-4 appears.



```
DBLD  IDS9.40.tar  images       README.html  SVR_ADM
[root@neon tmp]# ./ids_install

WARNING!
         This software, and its authorized use and number of users, are
subject to the applicable license agreement with IBM Corporation.
If the number of users exceeds the licensed number, the excess users may
be prevented from using the software.  UNAUTHORIZED USE OR COPYING MAY
SUBJECT YOU AND YOUR COMPANY TO SEVERE CIVIL AND CRIMINAL LIABILITIES.


Please Press RETURN to Continue, or (q/Q) to Quit :



IBM Informix Unix Bundle Installer

Installation Requirements:
- A user "informix" and a group "informix" must be known to the system.
- This installation procedure must be run by user root.
- Approximately 770 MB disk space required initially.
- Approximately 500 MB disk space if everything is installed (50 MB
  for ISA, 25 MB for either version of JDBC).

0) All Products listed below
1) IBM Informix Dynamic Server 9.40
2) IBM Informix IConnect
3) IBM Informix JDBC version 2.21
4) IBM Informix System Administrator
5) Configure a Demo IDS Server (requires IDS)

Enter the number(s) of the products to install, separated by spaces
(i.e. "1 2 3"):
```

Figure 2-4 IDS 9.40 installation menu

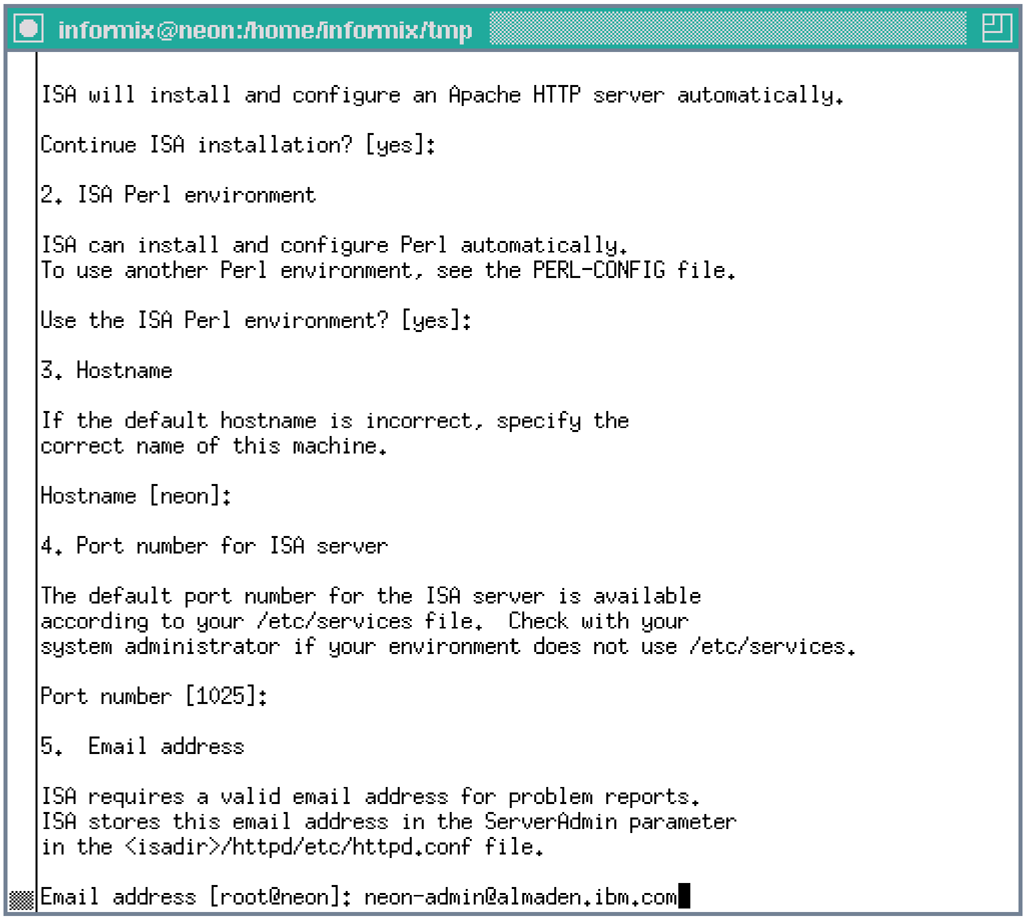
3. Choose option **0**, so all products are installed. Next the installation script will ask about the product directory (INFORMIXDIR). Since our INFORMIXDIR environment variable is set to **/home/IDS9.40** this is the default directory. Press **Enter** to continue and then specify the directory for the jdbc driver. In this case it should be a directory different than INFORMIXDIR. In our case we chose **/home/JDBC**, and the installation script created this directory and installed the jdbc files there. The menu shown in Figure 2-5 appears.



```
3) IBM Informix JDBC version 2.21
4) IBM Informix System Administrator
5) Configure a Demo IDS Server (requires IDS)

Enter the number(s) of the products to install, separated by spaces
(i.e. "1 2 3"):
0
Install into directory (INFORMIXDIR) [/home/IDS9.40] :

JDBC should not be installed in the same directory as other products. JDBC should
 only be installed in an empty directory or in a directory with a previous instal
lation of JDBC.
Install JDBC into directory (JDBCDIR) [/usr/jdbc] :
/home/JDBC
/bin/ls: /home/JDBC: No such file or directory
JDBCDIR does not exists. The JDBC installation will create this directory and con
tinue with the installation.

Installing System Administrator

13301 blocks
IBM Informix Server Administrator Installer


To use the default value that appears in [square brackets],
press the ENTER key.



1. ISA Apache HTTP server

ISA will install and configure an Apache HTTP server automatically.

Continue ISA installation? [yes]: 
```

Figure 2-5 Installation directory information

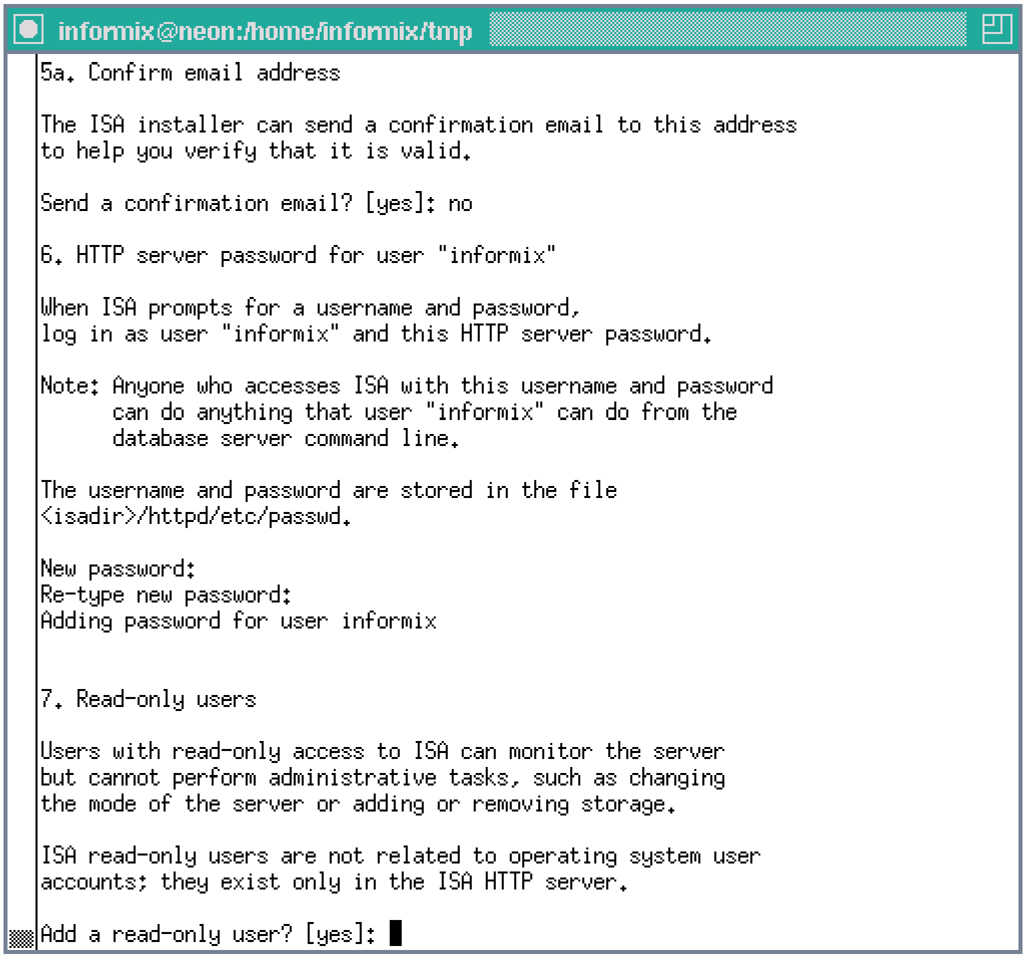
4. Next the installation script asks about the installation of the Informix Server Administrator (ISA) components, such as the Apache web server and the Perl environment. It also asks to confirm the hostname and choose a TCP/IP port for ISA. Press **Enter** for all questions, so everything will be installed and configured automatically. Finally it asks for an email address to be set in the Apache configuration file (httpd.conf). See Figure 2-6.



```
ISA will install and configure an Apache HTTP server automatically.

Continue ISA installation? [yes]:

2. ISA Perl environment

ISA can install and configure Perl automatically.
To use another Perl environment, see the PERL-CONFIG file.

Use the ISA Perl environment? [yes]:

3. Hostname

If the default hostname is incorrect, specify the
correct name of this machine.

Hostname [neon]:

4. Port number for ISA server

The default port number for the ISA server is available
according to your /etc/services file.  Check with your
system administrator if your environment does not use /etc/services.

Port number [1025]:

5.  Email address

ISA requires a valid email address for problem reports.
ISA stores this email address in the ServerAdmin parameter
in the <isadir>/httpd/etc/httpd.conf file.

Email address [root@neon]: neon-admin@almaden.ibm.com
```

Figure 2-6 Informix Server Administrator (ISA) components

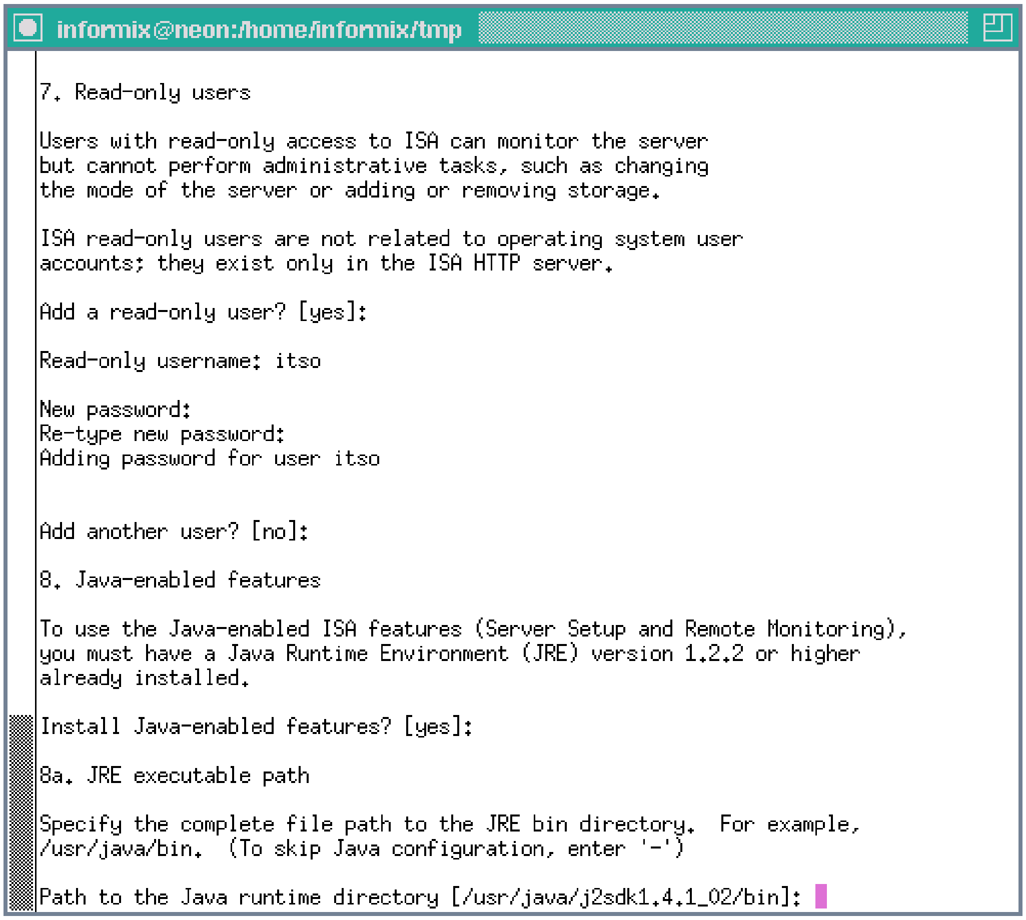
5. Next you are asked if you want a confirmation email to the address that you just entered. You will also be asked for a new password for ISA. See Figure 2-7.



```
5a. Confirm email address

The ISA installer can send a confirmation email to this address
to help you verify that it is valid.

Send a confirmation email? [yes]: no

6. HTTP server password for user "informix"

When ISA prompts for a username and password,
log in as user "informix" and this HTTP server password.

Note: Anyone who accesses ISA with this username and password
      can do anything that user "informix" can do from the
      database server command line.

The username and password are stored in the file
<isadir>/httpd/etc/passwd.

New password:
Re-type new password:
Adding password for user informix


7. Read-only users

Users with read-only access to ISA can monitor the server
but cannot perform administrative tasks, such as changing
the mode of the server or adding or removing storage.

ISA read-only users are not related to operating system user
accounts; they exist only in the ISA HTTP server.

Add a read-only user? [yes]:
```

Figure 2-7 Password for ISA

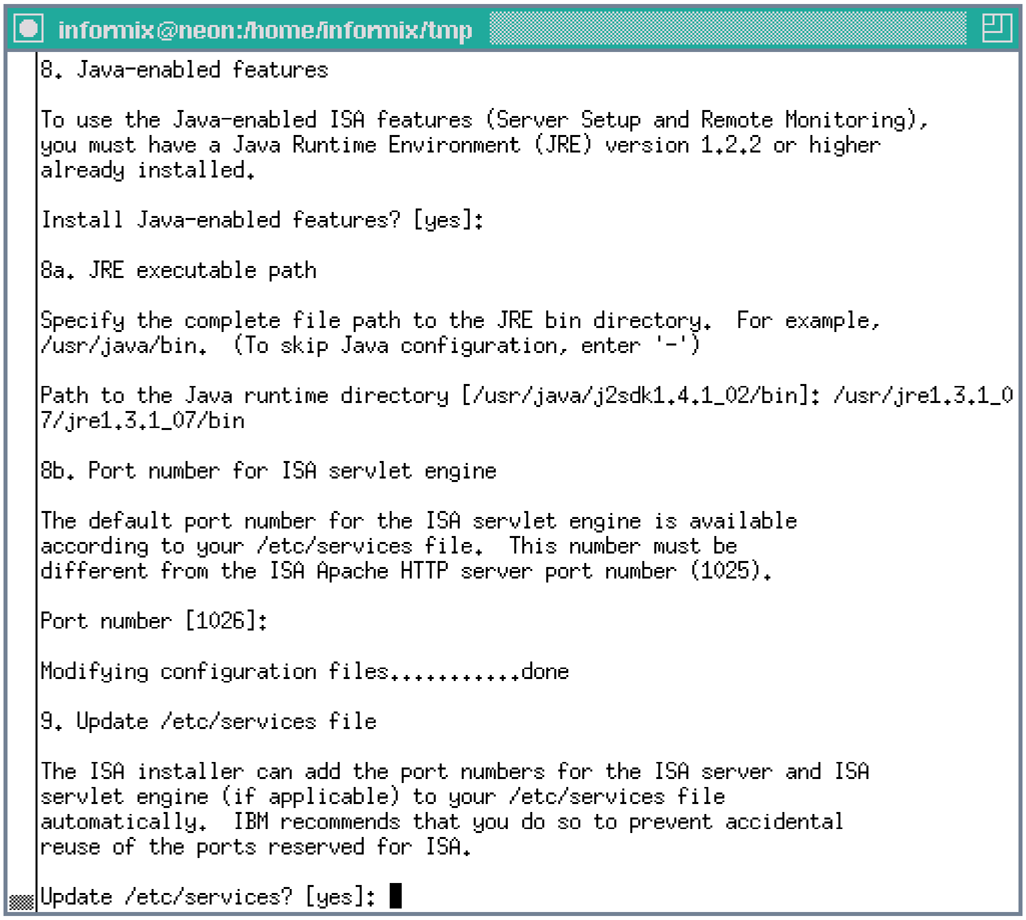
6. Here the installation script asks if you want to add read-only users to ISA. We added our generic user called **itso**, which is later used in some of the sample applications. See Figure 2-8.



```
7. Read-only users

Users with read-only access to ISA can monitor the server
but cannot perform administrative tasks, such as changing
the mode of the server or adding or removing storage.

ISA read-only users are not related to operating system user
accounts; they exist only in the ISA HTTP server.

Add a read-only user? [yes]:

Read-only username: itso

New password:
Re-type new password:
Adding password for user itso


Add another user? [no]:

8. Java-enabled features

To use the Java-enabled ISA features (Server Setup and Remote Monitoring),
you must have a Java Runtime Environment (JRE) version 1.2.2 or higher
already installed.

Install Java-enabled features? [yes]:

8a. JRE executable path

Specify the complete file path to the JRE bin directory.  For example,
/usr/java/bin.  (To skip Java configuration, enter '-')

Path to the Java runtime directory [/usr/java/j2sdk1.4.1_02/bin]:
```

Figure 2-8 Adding read-only users to ISA

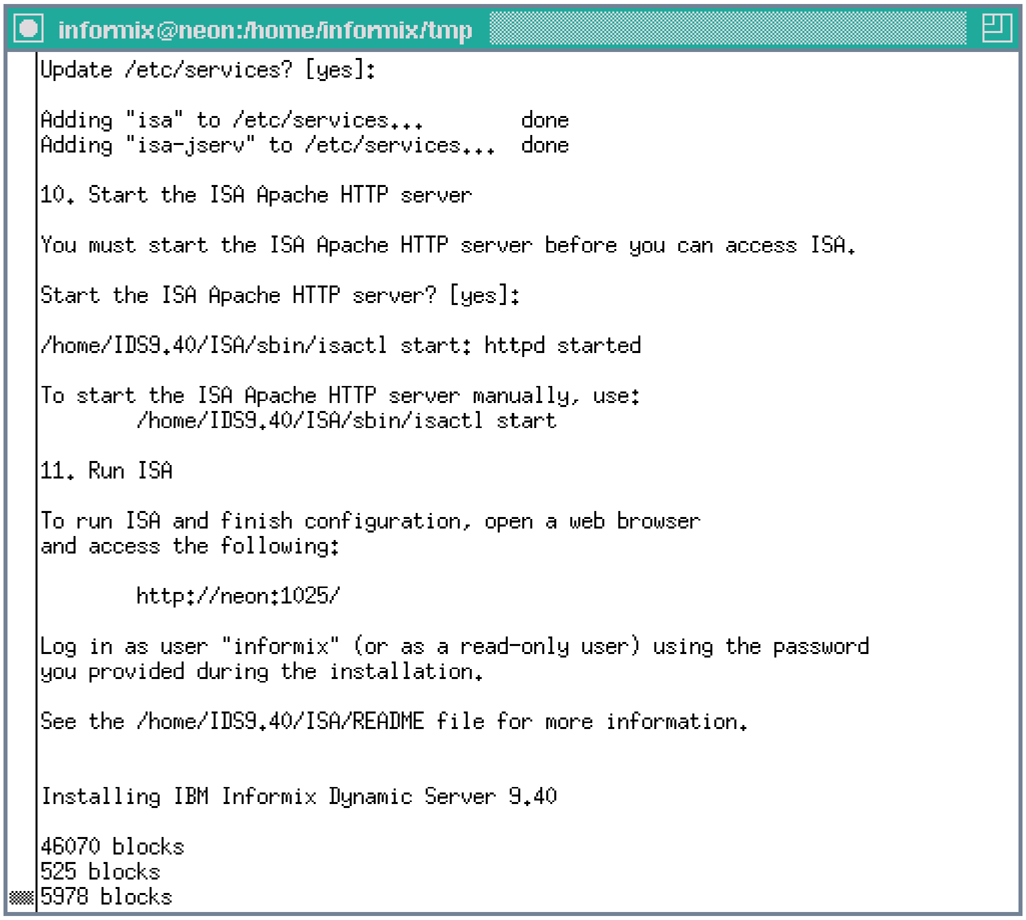
7. To use Java-enabled ISA features, you need to install JRE 1.2.2 or 1.3. In our case we had downloaded the JRE 1.3 from the Sun Web site at www.javasoft.com. A servlet engine is also installed to support this feature and the installation script asks for another TCP/IP port number. See Figure 2-9.



```
8. Java-enabled features

To use the Java-enabled ISA features (Server Setup and Remote Monitoring),
you must have a Java Runtime Environment (JRE) version 1.2.2 or higher
already installed.

Install Java-enabled features? [yes]:

8a. JRE executable path

Specify the complete file path to the JRE bin directory.  For example,
/usr/java/bin.  (To skip Java configuration, enter '-')

Path to the Java runtime directory [/usr/java/j2sdk1.4.1_02/bin]: /usr/jre1.3.1_0
7/jre1.3.1_07/bin

8b. Port number for ISA servlet engine

The default port number for the ISA servlet engine is available
according to your /etc/services file.  This number must be
different from the ISA Apache HTTP server port number (1025).

Port number [1026]:

Modifying configuration files...........done

9. Update /etc/services file

The ISA installer can add the port numbers for the ISA server and ISA
servlet engine (if applicable) to your /etc/services file
automatically.  IBM recommends that you do so to prevent accidental
reuse of the ports reserved for ISA.

Update /etc/services? [yes]:
```

Figure 2-9 Java-enabled ISA features installation

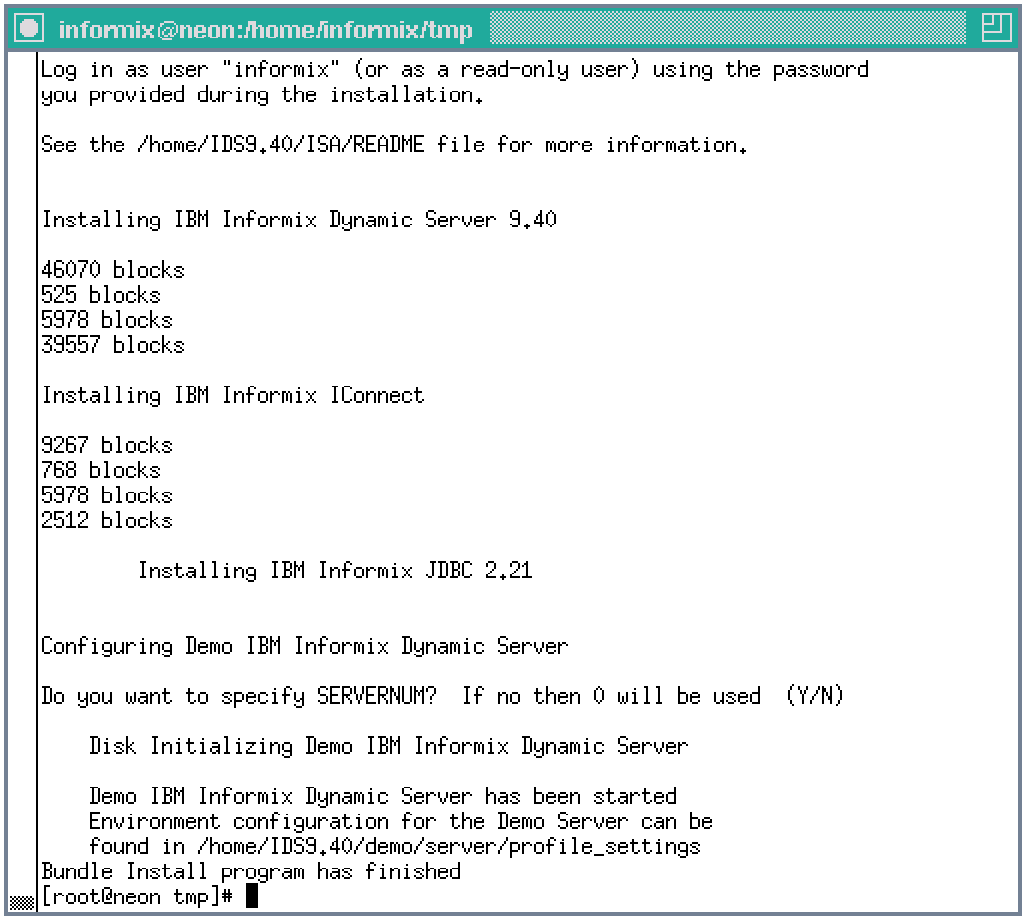
8. Next you are asked if the /etc/services file should be updated with the ISA TCP/IP ports. Enter **yes** for that. Then the installation script starts the Apache HTTP server configured for ISA and begins the installation of all products previously selected. See Figure 2-10.



```
Update /etc/services? [yes]:

Adding "isa" to /etc/services...        done
Adding "isa-jserv" to /etc/services...  done

10. Start the ISA Apache HTTP server

You must start the ISA Apache HTTP server before you can access ISA.

Start the ISA Apache HTTP server? [yes]:

/home/IDS9.40/ISA/sbin/isactl start: httpd started

To start the ISA Apache HTTP server manually, use:
        /home/IDS9.40/ISA/sbin/isactl start

11. Run ISA

To run ISA and finish configuration, open a web browser
and access the following:

        http://neon:1025/

Log in as user "informix" (or as a read-only user) using the password
you provided during the installation.

See the /home/IDS9.40/ISA/README file for more information.


Installing IBM Informix Dynamic Server 9.40

46070 blocks
525 blocks
5978 blocks
```

Figure 2-10 Starting ISA Apache server

9. IDS 9.40, I-Connect and the JDBC are installed. The installation script on IDS 9.40 provides a demo database server configuration. It creates an instance for you and updates all the configuration files. It also creates a file with the environment variables settings that you can include in the users profiles or execute it when accessing the database server. We used this demo server for our applications. See Figure 2-11.



```
Log in as user "informix" (or as a read-only user) using the password
you provided during the installation.

See the /home/IDS9.40/ISA/README file for more information.


Installing IBM Informix Dynamic Server 9.40

46070 blocks
525 blocks
5978 blocks
39557 blocks

Installing IBM Informix IConnect

9267 blocks
768 blocks
5978 blocks
2512 blocks

        Installing IBM Informix JDBC 2.21


Configuring Demo IBM Informix Dynamic Server

Do you want to specify SERVERNUM?  If no then 0 will be used  (Y/N)

    Disk Initializing Demo IBM Informix Dynamic Server

    Demo IBM Informix Dynamic Server has been started
    Environment configuration for the Demo Server can be
    found in /home/IDS9.40/demo/server/profile_settings
Bundle Install program has finished
[root@neon tmp]#
```

Figure 2-11 Installation of IDS 9.40, I-Connect and JDBC

10. The installation process is complete and an instance of the database server should be up and running. To verify that the installation was successful and that the demo instance is running do the following:

    `cd $INFORMIXDIR/demo/server`

    `cat ./profile_settings`

    INFORMIXSERVER=demo_on

    INFORMIXDIR=/home/IDS9.40

    ONCONFIG=onconfig.demo

    INFORMIXSQLHOSTS=/home/IDS9.40/etc/sqlhosts.std

    PATH=/home/IDS9.40/bin:/home/IDS9.40/bin:/home/IDS9.40/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin:/usr/java/j2sdk1.3.1_08/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/informix/bin:/home/informix/bin:/root/bin:/usr/java/j2sdk1.3.1_08/bin:/opt/netscape/netscape:.

    export INFORMIXSERVER INFORMIXDIR ONCONFIG INFORMIXSQLHOSTS PATH

    `. ./profile_settings`

    `onstat -`

You should then see the following:

Informix Dynamic Server Version 9.40.UC1 -- On-Line -- Up 00:10:42 -- 27920 Kbytes

You should see that the instance is On-Line,

Congratulations! You have successfully installed IDS 9.40 and configured a demo instance!

11. The demo applications that are developed throughout this book use the stores_demo database, so after the installation is complete we must create this demo database. Follow these steps:

    a. Log into the Linux server as user `itso`, so the database owner is the user `itso`.

    b. Since you will login as a new user you need to set the environment parameters to point to the correct server instance (**demo_on**). You might want to use the $INFORMIXDIR/demo/server/profile_settings file to do that.

    c. Run the following command to create the **stores_demo** database:

    ```
    dbaccessdemo -log
    ```

> **TIP:** During the database creation the IDS logical logs may fill up and the system will wait for a log backup. To avoid this situation in a test environment, you can change the LTAPEDEV configuration parameter to /dev/null. You can change this through ISA or by editing the $ONCONFIG file.

12. We also need to make sure that the TCP/IP connection protocol is enabled in our server instance, since we are going to have WebSphere connect to IDS from a remote server.

    The best way to verify that is through the environment variables, onconfig, and sqlhosts files. You might also need to check OS files, such as /etc/services:

    a. The first thing to verify is your INFORMIXSERVER environment variable:

    ```
    [informix@neon IDS9.40]$ echo $INFORMIXSERVER
    demo_on
    ```

    b. Verify that this is also the value in your server configuration file (onconfig). The value demo_on should also be configured in either the DBSERVERNAME or DBSERVERALIASES parameters:

    ```
    [informix@neon IDS9.40]$ echo $ONCONFIG
    onconfig.demo
    [informix@neon IDS9.40]$ grep DBSERVER $INFORMIXDIR/etc/$ONCONFIG
    DBSERVERALIASES                  # List of alternate dbservernames
    DBSERVERNAME demo_on
    ```

    c. Now verify that demo_on uses a TCP/IP protocol in the sqlhosts file. The column in question to verify that is the second one, which specifies the NETTYPE value. On Linux it should be *onsoctcp*. First find the sqlhosts file. If you are not setting the INFORMIXSQLHOSTS environment variable, the sqlhosts file defaults to $INFORMIXDIR/etc/sqlhosts. In our case we use $INFORMIXSQLHOSTS.

    ```
    [informix@neon IDS9.40]$ cat $INFORMIXSQLHOSTS
    #**************************************************************************
    #   Title:      sqlhosts.demo
    #   Sccsid:     @(#)sqlhosts.demo       9.2     7/15/93  15:20:45
    ```

```
#   Description:
#             Default sqlhosts file for running demos.
#************************************************************************
demo_on       onsoctcp       neon    demo_on_tcp
```

d. Based on the sqlhosts file we know that the TCP/IP port name is demo_on_tcp (forth column). To get the exact port number, look for this name in the /etc/services file:

```
[informix@neon IDS9.40]$ grep demo_on_tcp /etc/services
demo_on_tcp       1533/tcp
```

Finally we verify that our server instance has the TCP/IP protocol enabled and that it uses port number 1533.

Note: If your server instance still uses only the shared memory protocol (onipcshm), make the following changes:

1. Take the server offline: onmode -ky
2. Edit the sqlhosts file and edit the line that corresponds to your INFORMIXSERVER value. Change the NETTYPE value (second column), from onipcshm to onsoctcp
3. Include the value specified in the forth column in the /etc/services file and assign an unique port number to it.
4. Bring the server online: oninit

Configuring ISA to manage the demo instance

During the IDS 9.40 installation process we chose Informix Server Administrator (ISA) to be installed as well. ISA is a web based application that is used to manage IDS instances, either locally or remotely. Here we show how to configure it to manage the demo instance.

1. After finishing the installation open a web browser and use this URL format:

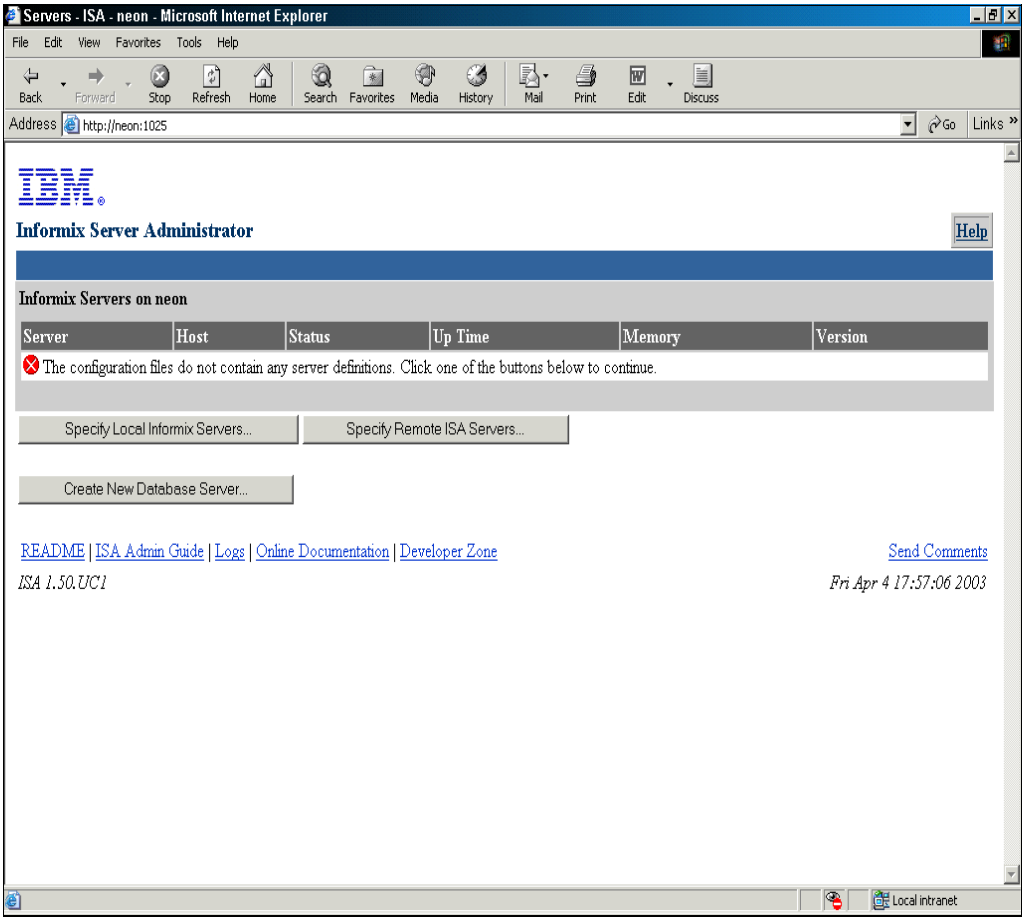
   http://<host>:<ISA port number>

   For example, in our case this is:

   ```
   http://neon:1025
   ```

   Note: On step 4 of the installation process we have to specify a TCP/IP port number for ISA. In our case we chose the default port, 1025.

   The page shown in Figure 2-12 should be loaded.


Servers - ISA - neon - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://neon:1025
Go Links
IBM
Informix Server Administrator
Help
Specify Local Informix Servers...
Specify Remote ISA Servers...
Create New Database Server...
README | ISA Admin Guide | Logs | Online Documentation | Developer Zone
Send Comments
Local intranet

Informix Servers on neon

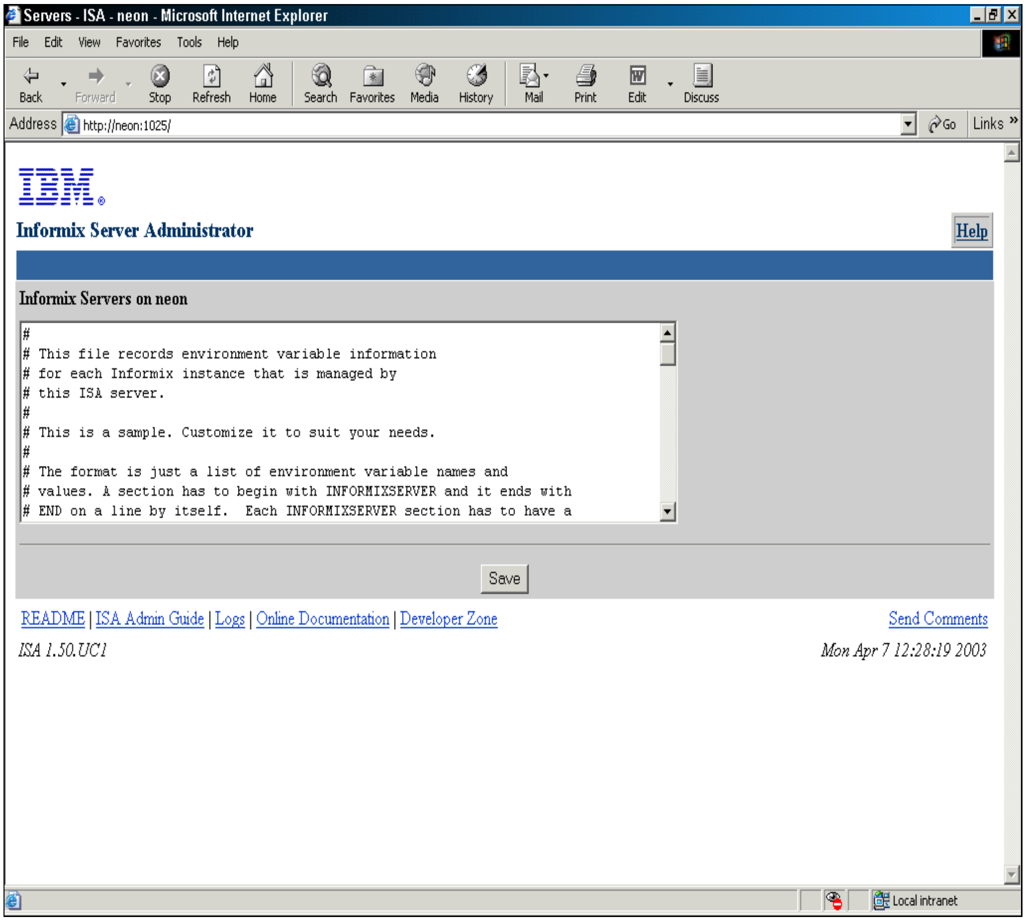
Server	Host	Status	Up Time	Memory	Version
☒ The configuration files do not contain any server definitions. Click one of the buttons below to continue.					

ISA 1.50.UC1

Fri Apr 4 17:57:06 2003

Figure 2-12 ISA page

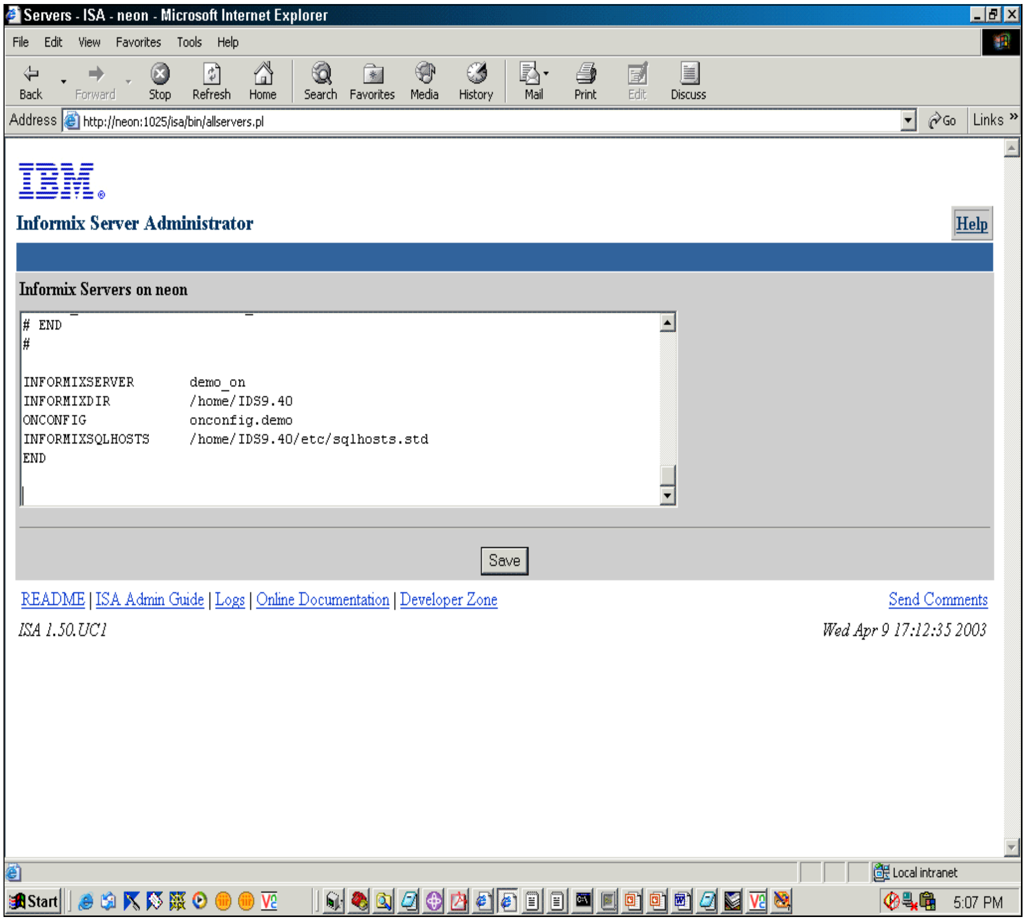
2. You will notice that this page does not show any server instance configured for ISA. We need to specify some parameters to have ISA manage our demo instance. Click the **Specify Local Informix Servers...** button, and you should get the screen shown in Figure 2-13.


Servers - ISA - neon - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://neon:1025/
Go Links
IBM
Informix Server Administrator
Help
Informix Servers on neon
Save
README | ISA Admin Guide | Logs | Online Documentation | Developer Zone
Send Comments
ISA 1.50.UC1
Mon Apr 7 12:28:19 2003
Local intranet

```
#
# This file records environment variable information
# for each Informix instance that is managed by
# this ISA server.
#
# This is a sample. Customize it to suit your needs.
#
# The format is just a list of environment variable names and
# values. A section has to begin with INFORMIXSERVER and it ends with
# END on a line by itself.  Each INFORMIXSERVER section has to have a
```

Figure 2-13 Configuration screen

3. Scroll down and follow the configuration instructions. Basically to configure a new local instance in ISA we just have to set the environment variables that identify the server. Go to the end of the configuration file and add your parameters. In our case the values (same values as configured in $INFORMIXDIR/demo/server/profile_settings) are shown in Figure 2-14.


Servers - ISA - neon - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://neon:1025/isa/bin/allservers.pl
Go Links
IBM
Help
Local intranet
Start
5:07 PM

Informix Server Administrator

Informix Servers on neon

```
# END
#

INFORMIXSERVER       demo_on
INFORMIXDIR          /home/IDS9.40
ONCONFIG             onconfig.demo
INFORMIXSQLHOSTS     /home/IDS9.40/etc/sqlhosts.std
END
```

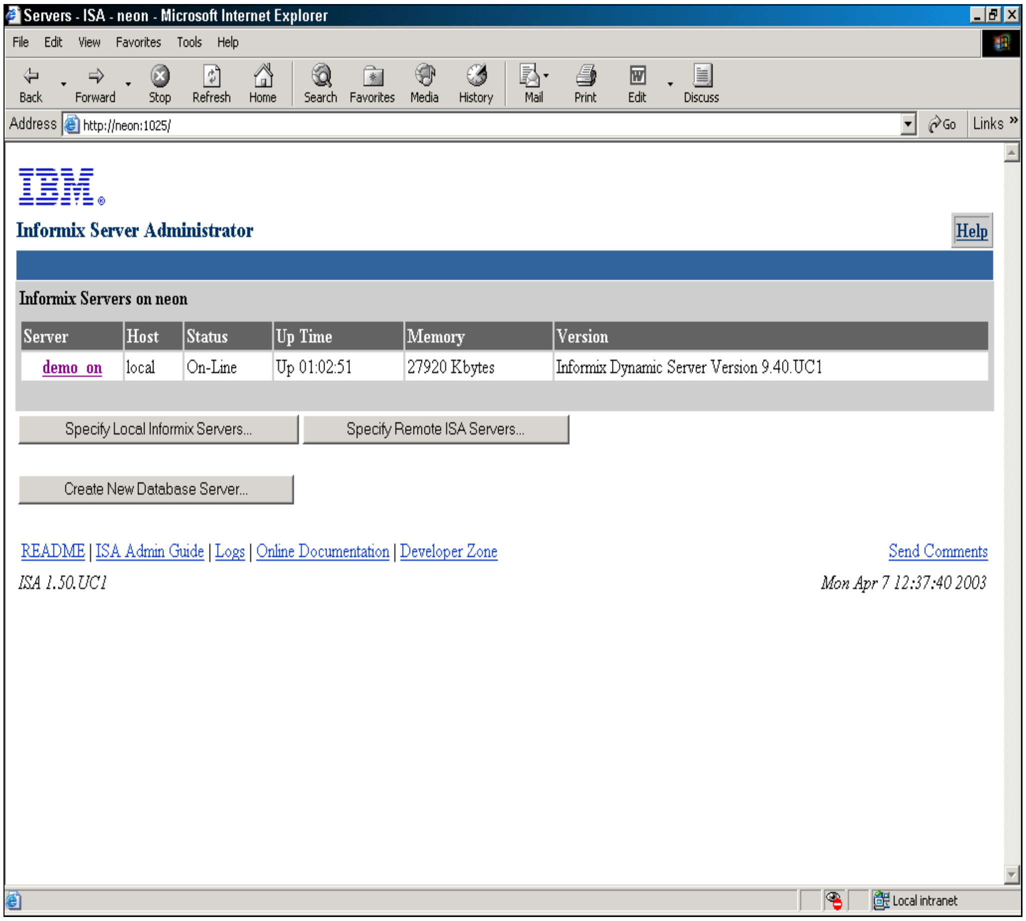
Save

README | ISA Admin Guide | Logs | Online Documentation | Developer Zone   Send Comments

ISA 1.50.UC1   Wed Apr 9 17:12:35 2003

Figure 2-14 Specifying instance environment variables

Click the **Save** button and you should see the **demo_on** server configured, as shown in Figure 2-15.


Servers - ISA - neon - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://neon:1025/
Go Links
IBM
Help
Specify Local Informix Servers...
Specify Remote ISA Servers...
Create New Database Server...
README | ISA Admin Guide | Logs | Online Documentation | Developer Zone
Send Comments
Local intranet

Informix Server Administrator

Informix Servers on neon

Server	Host	Status	Up Time	Memory	Version
demo_on	local	On-Line	Up 01:02:51	27920 Kbytes	Informix Dynamic Server Version 9.40.UC1

ISA 1.50.UC1

Mon Apr 7 12:37:40 2003

Figure 2-15 Demo instance configured to be managed by ISA

4. ISA is now ready to use. Click the server name (**demo_on**) to open the monitoring and configuration page of this instance. The first page shows the IDS log file (online.log) and on the left frame you see all the options available in ISA to manage the server instance, as shown in Figure 2-16.

5. With ISA you can perform the following tasks:

   – Change the server mode (online, quiescent, offline). The quiescent mode is the administrative mode of IDS, where only the user informix has access to the database server.

   – Check and change configuration parameters.

   – Monitor logical and physical logs, memory, users, etc.

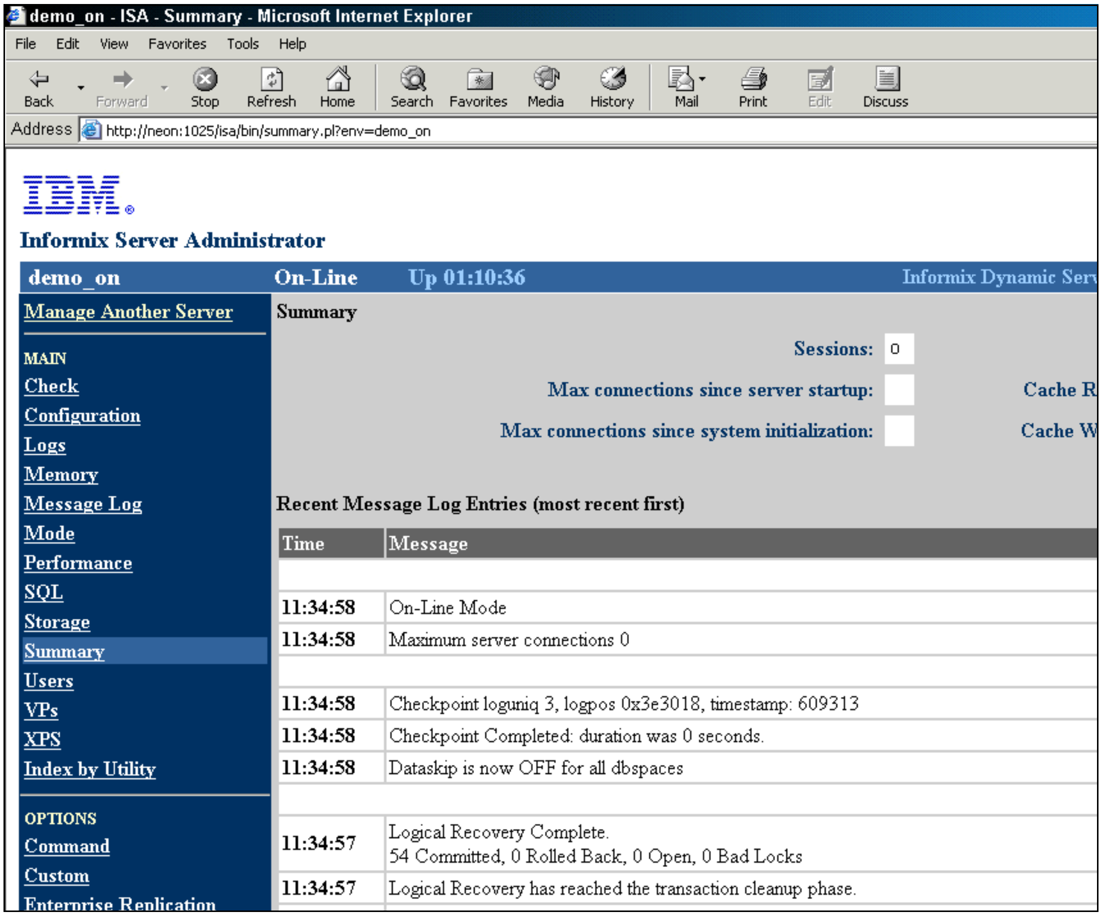
   – Check the database integrity.

   – Run shell commands.

   – Configure and monitor ER.

   – Configure and monitor backup utilities, like ISM and ON-Bar.

   – Create a new server instance.


demo_on - ISA - Summary - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://neon:1025/isa/bin/summary.pl?env=demo_on
IBM
Informix Server Administrator
demo_on On-Line Up 01:10:36 Informix Dynamic Serv
Manage Another Server
MAIN
Check
Configuration
Logs
Memory
Message Log
Mode
Performance
SQL
Storage
Summary
Users
VPs
XPS
Index by Utility
OPTIONS
Command
Custom
Enterprise Replication

Summary

Sessions: 0

Max connections since server startup:

Max connections since system initialization:

Cache R

Cache W

Recent Message Log Entries (most recent first)

Time	Message
11:34:58	On-Line Mode
11:34:58	Maximum server connections 0
11:34:58	Checkpoint loguniq 3, logpos 0x3e3018, timestamp: 609313
11:34:58	Checkpoint Completed: duration was 0 seconds.
11:34:58	Dataskip is now OFF for all dbspaces
11:34:57	Logical Recovery Complete. 54 Committed, 0 Rolled Back, 0 Open, 0 Bad Locks
11:34:57	Logical Recovery has reached the transaction cleanup phase.

Figure 2-16 Main page for managing the demo instance

Command-line option to manage IDS 9.40

In the last section we showed how to configure ISA to manage the administration tasks of IDS. Even though this is a very good option (GUI oriented, easy to use, web tool), many informix administrators still prefer to use the command line utilities. Here we show the most common utilities and some of their options. Some of the utilities have to be run as user informix.

Important: All the utilities are located in the $INFORMIXDIR/bin directory, so make sure to include it in the PATH environment variable.

oninit

This utility is used to initialize the server. It can either initialize only the shared memory of a configured instance or it can be used to initialize both the disk and the shared memory for a new instance (**-i** option).

Example 2-7 and Example 2-8 show these oninit options.

Example 2-7 Bringing the server to online mode

```
/* If the server is offline this command brings it to online mode: */
[informix@neon bin]$ oninit
```

Example 2-8 Initialize shared memory and disk structures

```
/* The next command initializes both the disk and the shared memory: */
[informix@neon bin]$ oninit -i
```

Attention: The **-i** option of the oninit utility has to be used with care, because it can reinitialize the disk and all data stored in the databases is lost!! Only use this option if you are configuring a new instance or if you do not care about the data stored in the existing instance.

onmode

The onmode utility is used for many administrative tasks, such as:

- Change the server mode. We use onmode to change IDS to all modes except online, which is performed using the oninit utility.
- Kill server sessions and transactions.
- Add or remove resources to the database server, like shared memory segments or virtual processors. This task is performed dynamically with the server in online mode.
- onmode is also used to downgrade the server to an earlier version.
- Controls server mode (primary, secondary, standard) of a HDR server

Example 2-9, Example 2-10, and Example 2-11 show some of the onmode options.

Example 2-9 Changes server to maintenance mode (quiescent)

```
/* This option changes the server mode to quiescent. It waits for the users to finish their
work, so it is also referred as a graceful shutdown. If you don’t want to wait for the
users you can use the -u option */
[informix@neon bin]$ onmode -s
```

Example 2-10 Take server offline

```
/* The following option take server offline without waiting for users and answering yes to
all questions used in the -k option. */
[informix@neon bin]$ onmode -ky
```

Example 2-11 Adding a new VP dynamically

```
/* You can add a new VP to the server using onmode. For instance, to add a new CPU vp: */
[informix@neon bin]$ onmode -p +1 CPU
```

oncheck

The onchek utility is basically used to accomplish three tasks: to check disk structures for inconsistencies (corruptions), to repair indexes, or to display information about disk structures.

Example 2-12 and Example 2-13 show some of these oncheck options.

Example 2-12 Checking data and index consistency

```
/* Command to check the consistency of data and index pages on database wsad5: */
[informix@neon bin]$ oncheck -cDI wsad5
```

Example 2-13 Showing detailed table information

```
/* Command to display the information about a table on database wsad5 */
[informix@neon bin]$ oncheck -pt wsad5:customer
TBLspace Report for wsad5:itsobank.customer

    Physical Address               1:83
    Creation date                  04/07/2003 11:09:27
    TBLspace Flags                 901        Page Locking
                                              TBLspace contains VARCHARS
                                              TBLspace use 4 bit bit-maps
    Maximum row size               85
    Number of special columns      2
    Number of keys                 0
    Number of extents              1
    Current serial value           1
    First extent size              8
    Next extent size               8
    Number of pages allocated      8
    Number of pages used           2
    Number of data pages           1
    Number of rows                 11
    Partition partnum              1048646
    Partition lockid               1048646

    Extents
         Logical Page     Physical Page        Size
                    0           1:9651            8

                  Index  100_1 fragment in DBspace rootdbs

    Physical Address               1:84
    Creation date                  04/07/2003 11:09:27
    TBLspace Flags                 801        Page Locking
                                              TBLspace use 4 bit bit-maps
    Maximum row size               85
    Number of special columns      0
    Number of keys                 1
    Number of extents              1
    Current serial value           1
    First extent size              4
    Next extent size               4
    Number of pages allocated      4
    Number of pages used           4
    Number of data pages           0
    Number of rows                 0
    Partition partnum              1048647
    Partition lockid               1048646

    Extents
         Logical Page     Physical Page        Size
                    0           1:9659            4
```

onstat

The onstat is the major utility to monitor database activities, display shared memory structures, and show statistics for performance tuning purposes.

> **Note:** The onstat command has a subset of options called MT options. In order to use the MT commands you also have to specify the **-g** option.

Example 2-14, Example 2-15, and Example 2-16 show these onstat options.

Example 2-14   Checking disk space (dbspaces and chunks)

```
/* Command to show chunks and dbspaces information: */
[informix@neon bin]$ onstat -d
Informix Dynamic Server Version 9.40.UC1     -- On-Line -- Up 00:28:50 -- 27920 Kbytes

Dbspaces
address  number   flags      fchunk   nchunks  flags    owner    name
0x30c887d8 1        0x1        1        1        N       informix rootdbs
 1 active, 2047 maximum

Chunks
address  chunk/dbs  offset     size       free       bpages     flags pathname
0x30c88928 1     1    0          15000      5189                PO--
/home/IDS9.40/demo/server/online_root
 1 active, 2047 maximum

Expanded chunk capacity mode: disabled
```

Example 2-15   Showing users sessions

```
/* Command to show user sessions (MT option): */
[informix@neon bin]$ onstat -g ses
Informix Dynamic Server Version 9.40.UC1     -- On-Line -- Up 00:01:04 -- 27920 Kbytes

session                                      #RSAM    total      used       dynamic
id       user     tty      pid      hostname threads  memory     memory     explain
22       informix -        0        -        0        12288      7888       off
21       itsobank -        -1       dhcp3910 1        45056      30344      off
20       eduardo  3        1566     neon     1        36864      29888      off
4        itsobank 2        1440     neon     1        36864      29888      off
3        informix -        0        -        0        12288      9096       off
2        informix -        0        -        0        12288      7888       off
```

Example 2-16   Showing profile of activity on the server

```
/* Command to show profile of activity. This option is used mostly to tune the server */
[root@neon server]# onstat -p
Informix Dynamic Server Version 9.40.UC1     -- On-Line -- Up 00:03:30 -- 27920 Kbytes
Profile
dskreads pagreads bufreads %cached dskwrits pagwrits bufwrits %cached
323      454      45373    99.29   164      869      15085    98.91

isamtot  open     start    read     write    rewrite  delete   commit   rollbk
32252    3982     4718     7516     3166     537      53       151      1

gp_read  gp_write gp_rewrt gp_del   gp_alloc gp_free  gp_curs
0        0        0        0        0        0        0

ovlock   ovuserthread ovbuff   usercpu  syscpu   numckpts flushes
0        0            0        3.06     0.42     2        4

bufwaits lokwaits lockreqs deadlks  dltouts  ckpwaits compress seqscans
11       0        12159    0        0        0        1020     91

ixda-RA  idx-RA   da-RA    RA-pgsused lchwaits
11       0        0        11         0
```

There are many other utilities that are used to mange an Informix instance, such as onmonitor, onspaces, dbexport/dbimport, onparams, onlog, ontape, and onbar. We do not show all of them here, if you want more information about the command line utilities see the *Informix Dynamic Server Administrator's Reference manual* (G251-1250-00).

Data administration on IDS

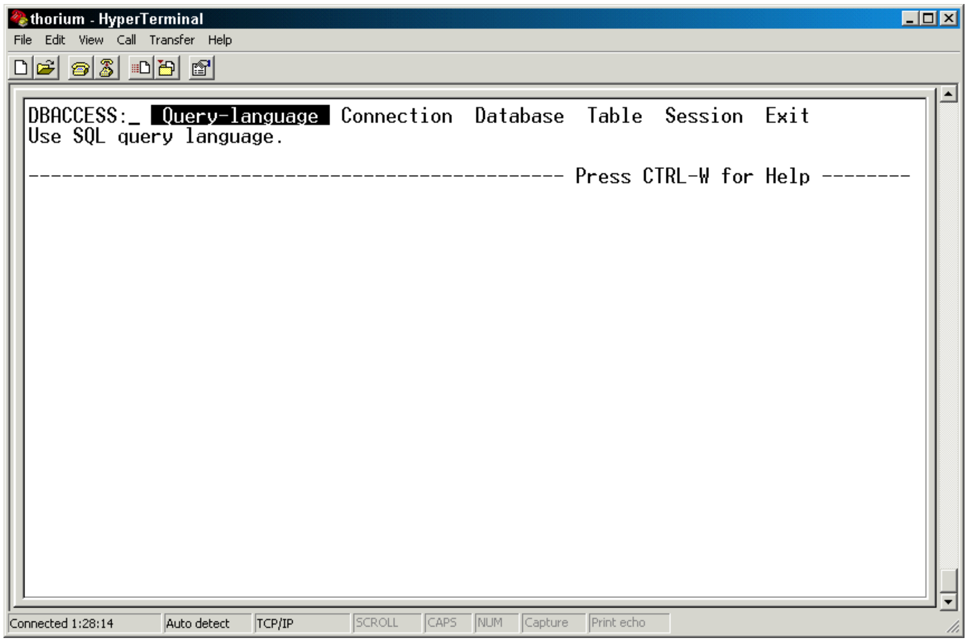
In the last sections we showed how the database server administrator can manage IDS using either ISA or the command-line utilities. In this section we show how a data administrator (sometimes it is the same person that administer the server) and developers can manage the database objects (databases, tables, indexes, constraints) and run SQL statements.

There are two primary tools for running and managing database objects: You can either use a character-based tool called dbaccess, or a third-party GUI tool called Server Studio. Dbaccess is installed during the IDS installation process. However if you prefer to use Server Studio you will need to download and install it. Here we show an overview of both tools.

Dbaccess

Dbaccess is a character-based tool that is installed along with the database server, and for many server and data administrators it is the preferred tool to manage data and run SQL commands.

The dbaccess binary resides in the $INFORMIXDIR/bin directory, so if it is configured in the PATH environment, you can just type **dbaccess** and a menu-driven application is loaded, as shown in Figure 2-17.



```
DBACCESS:_  Query-language  Connection  Database  Table  Session  Exit
Use SQL query language.

------------------------------------------------ Press CTRL-W for Help --------
```

Figure 2-17 Dbaccess first screen

The menu provides five options, and most of them have sub-menus with related options. Here are some brief descriptions of these options:

- **Query-Language:** Use this option to run SQL statements.
- **Connection:** This option is used to connect to either local or remote databases that are configured in the sqlhosts file.

- **Database:** This option is used to manage databases. You can create a new database on this server, drop an existing database, select a database from a list, get information, or close a selected database.
- **Table:** Use this option to manage a table in a selected database. You can create a new table, alter or drop an existing one, or get information about it.
- **Session:** Retrieve information about the current dbaccess session

For example, to run a select statement on the customers table, follow these steps:

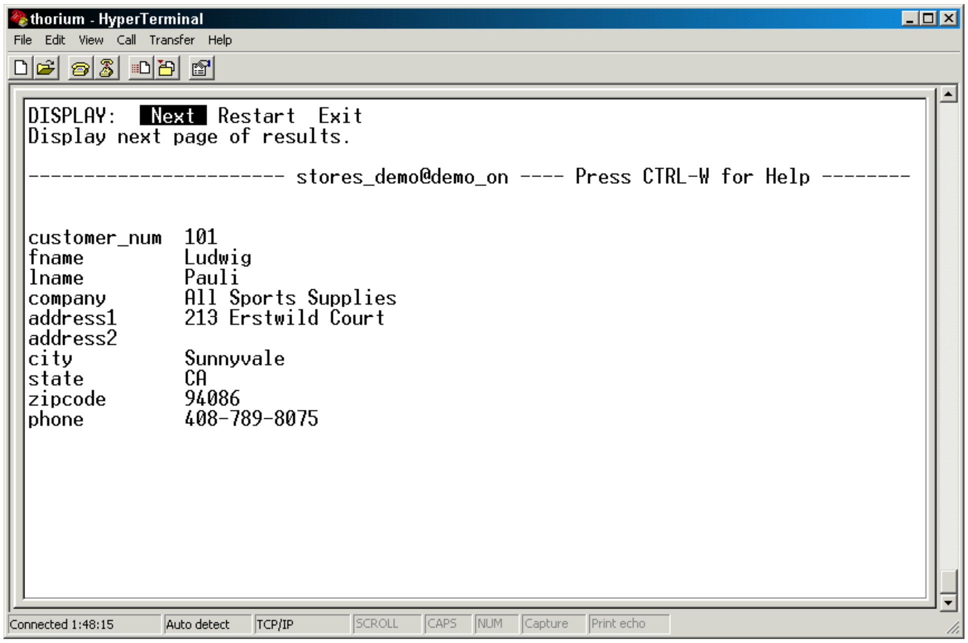
1. Open a terminal window on the database server using either the console or a telnet session. The **stores_demo** was previously created as user **itso**, so log into the server using this user id.
2. If not set, configure the PATH environment variable:

   ```
   export PATH=$INFORMIXDIR/bin:$PATH
   ```

3. Run **dbaccess**, choose **Query-Language** and select the **stores_demo** database
4. Select New and specify the following SQL command:

   ```
   select * from customer
   ```

5. Press the **ESC** button when you are finished editing and choose **Run** to execute the query. Figure 2-18 shows the expected result.



```
DISPLAY:  Next  Restart  Exit
Display next page of results.

---------------------- stores_demo@demo_on ---- Press CTRL-W for Help --------


customer_num  101
fname         Ludwig
lname         Pauli
company       All Sports Supplies
address1      213 Erstwild Court
address2
city          Sunnyvale
state         CA
zipcode       94086
phone         408-789-8075
```

Figure 2-18 Retrieving data from the customers table

> **TIP:** You may have problems trying to navigate on dbaccess using the arrow keys. This is probably related to your terminal settings. To correct the problem, set the TERM and TERMCAP environment variables as follows:
>
> ```
> export TERM=vt100
>
> export TERMCAP=$INFORMIXDIR/etc/termcap
> ```

For more information about the dbaccess utility, see the *IBM Informix DB-Access User's Guide, Version 9.40* (G251-1239-00).

AGS Server Studio

Server Studio is a GUI administration tool provided by AGS. The basic modules of the tool are: SQL editor, object explorer, and the table editor.

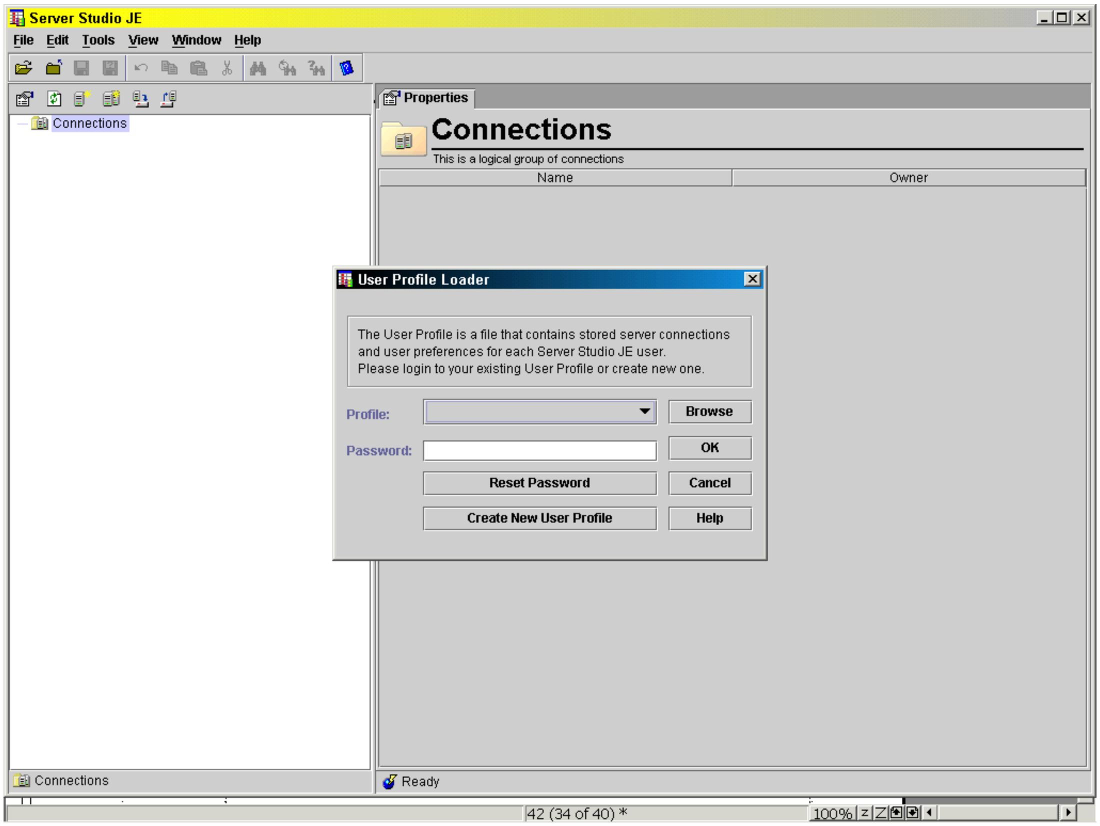
Server Studio is available for download (Java and Win32 versions are available) at:

http://www.serverstudio.com/downloads.html

On the same URL you will find the system requirement and also the installation instructions for Server Studio (we used the Java version). Once Server Studio is installed in your chosen directory, we just have to launch the program:

```
C:\ServerStudioJE>ServerStudioJE
```

This is what you should see (Figure 2-19).



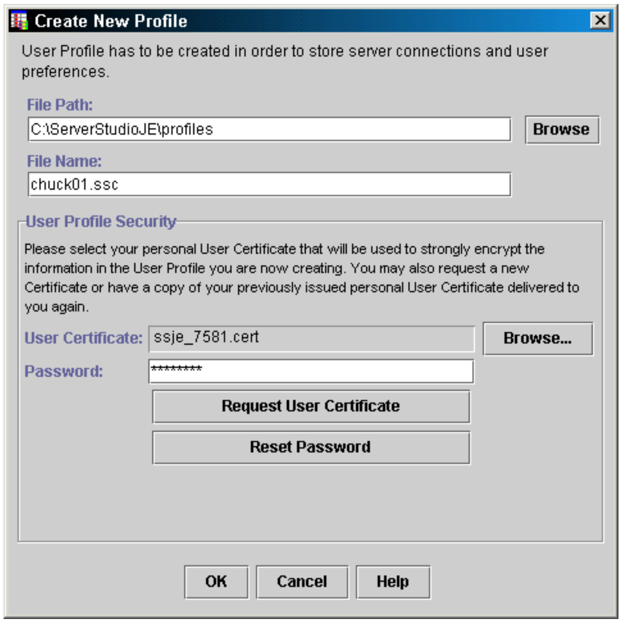
This is a logical group of connections

The User Profile is a file that contains stored server connections and user preferences for each Server Studio JE user.
Please login to your existing User Profile or create new one.

Figure 2-19 Server Studio - Profile validation

Before using Server Studio you need to create a new user profile and request a new user certificate. Basically you have to register as a new AGS user and a certification file (used with a password authentication) is mailed to you via email. Just follow the screen instructions to obtain this file.

Once you receive the certification file you can continue creating your new profile, as shown in Figure 2-20.



Create New Profile

User Profile has to be created in order to store server connections and user preferences.

File Path: C:\ServerStudioJE\profiles

File Name: chuck01.ssc

User Profile Security

Please select your personal User Certificate that will be used to strongly encrypt the information in the User Profile you are now creating. You may also request a new Certificate or have a copy of your previously issued personal User Certificate delivered to you again.

User Certificate: ssje_7581.cert

Password: *********

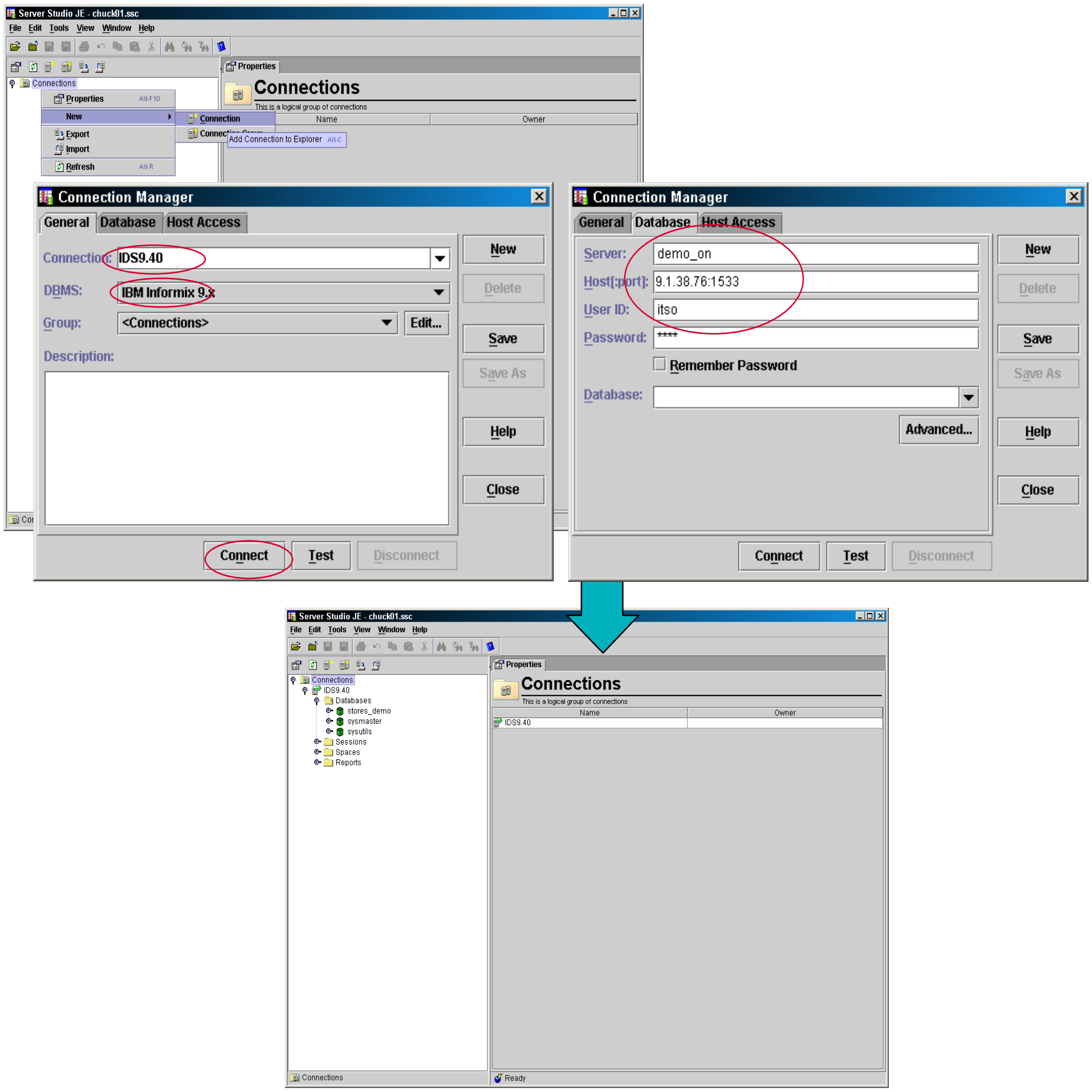
Figure 2-20 Creating new profile

After the profile is created the NetUpdate window is displayed. NetUpdate is an agent that checks for new versions of Server Studio that are available for an update. On this window you can set the frequency of this checking procedure. We chose to have a version checking now.

After the version update you are ready to explore the databases objects on IDS 9.40. First you need to create and configure a connection to the server:

To do this we will be referring to Figure 2-21. To configure use the following steps:

1. Right-click the connection frame and select **New -> Connection,** and then connection window appears.
2. In the connection manager window choose a name for your connection. These are highlighted with an ellipse around them. We chose **IDS9.40**. Then for DBM, choose **IBM Informix 9.x.**
3. Now select the database. Here we must configure the server instance to which we will connect, the host where the instance resides, and the user id and password. In our case we used, **demo_on, 9.1.38.76, 1533,** and **itso.** These are highlighted with a large ellipse. Although not shown on the figure, we need to enter the database name, which in our case was **stores_demo**.
4. Click the **Connect** button to verify the configuration.



Connection: IDS9.40

DBMS: IBM Informix 9.x

Group: <Connections>

Description:

Server: demo_on

Host[:port]: 9.1.38.76:1533

User ID: itso

Password: ****

☐ Remember Password

Database:

Figure 2-21 Configuring connection to IDS

Now you can control all database objects of this connection (provided that you have the necessary privileges). You can access such elements as tables, views, and triggers. You can also run SQL commands, import and export data, and run update statistics.

AGS also provides add-on modules that helps you have better control over your data and even administer the server (as a option for ISA). For more information about AGS server studio, check the following site:

http://www.serverstudio.com/

2.3 Configuring IDS for use with WebSphere

In this section we describe the procedure to configure the IDS 9.40 environment to be integrated with WebSphere. It is not our intention to show how to tune IDS 9.40 to work on a OLTP or DSS system (although we do provide some general guidelines in the chapter on Hints and Tips), for this kind of configuration we recommend specialized performance and tuning publications. The objective here is to show the basic configuration that needs to be performed in order to enable IDS to work with WebSphere products.

In this book we use WebSphere Studio Application Developer 5.0 on Windows/2000, and WebSphere Application Server 5.0 on a Linux server. These two products are described in the following chapters. But, it is important to note that the WebSphere family constitutes the IBM solution for an e-business infrastructure. It is also important to note that WebSphere is based on the J2EE industry standard, and uses various Java technologies. What that means is, that from the IDS standpoint, we need to configure IDS to work with Java applications. Java programs need to be able to connect to the database server, run queries and execute transactions. This is accomplished through Java database connectivity (JDBC).

JDBC and the JDBC drivers

Java database connectivity (JDBC) is the JavaSoft specification of a standard application programming interface (API) that allows Java programs to access database management systems. The JDBC API consists of a set of interfaces and classes written in the Java programming language. Using these standard interfaces and classes, programmers can write applications that connect to databases, send queries written in structured query language (SQL), and process the results.

The JDBC API defines the Java interfaces and classes that programmers use to connect to databases and send queries. A JDBC driver implements these interfaces and classes for a particular DBMS vendor. There are four types of JDBC drivers:

- JDBC-ODBC bridge plus ODBC driver, also called Type 1.
- Native-API, partly Java driver, also called Type 2.
- JDBC-Net, pure-Java driver, also called Type 3.
- Native-protocol, pure-Java driver, also called Type 4.

The Informix JDBC driver is a native-protocol, pure-Java driver (Type 4). A type 4 JDBC driver provides direct connection to the database server, without a middle tier. The informix JDBC driver version that we use in this book is 2.21.JC4 and it is based on version 2.0 of the JDBC API.

Installation and configuration of Informix JDBC driver

The JDBC driver was installed along with IDS 9.40 in Section , “Installing” on page 33. However, note that some IDS versions do not come with JDBC bundled in the distribution files. So, if you need to download the Informix JDBC driver, go to the following URL:

```
http://www-3.ibm.com/software/data/informix/
```

On the left frame, click the **Downloads** link, and it redirects you to the download page of Informix products. In the search field only the keyword **Informix** is specified. Add the keyword **jdbc** and select the **all keywords button**. This will filter the results and make the search more effective.

Our JDBC driver was previously installed in the /home/JDBC directory (Step 3 on page 34). The configuration requirement is to set the CLASSPATH environment variable and include the JDBC jar files in it. For example, in our case:

```
export CLASSPATH=/home/JDBC/lib/ifxjdbc.jar:/home/JDBC/lib/ifxjdbcx.jar
```

TIP: Set the CLASSPATH environment variable in every user profile that will connect to IDS using JDBC. In our case we added this variable in the profile_settings file and loaded this script in the user profiles that ran Java applications against IDS 9.40.

By including the Informix JDBC driver in the CLASSPATH we make available the necessary packages that a Java application needs to access to connect to an Informix server. Inside the application code we have to import the JDBC packages, load the JDBC driver, and then connect to the database.

Configuration checking using a sample application

The best way to test the IDS and JDBC configuration is to execute a simple Java application that connects to IDS using a JDBC connection. If this works fine we verify that the IDS environment is ready to be integrated with WebSphere.

In this section we use a sample Java application that loads the JDBC driver, uses a connection URL to specify configuration parameters, and connects to the stores_demo database. If it connects successfully data is retrieved from the customer table.

Before running this demo you need to follow these steps:

1. Install a Java environment package in your Linux server. In our case we downloaded a Java 2 Platform, Standard Edition (J2SE) version 1.3.1 from http://www.java.sun.com/j2se/downloads.html
2. Include the bin directory of J2SE in the PATH environment variable (Java on Linux servers is usually installed in the /usr directory):

   ```
   export PATH=$PATH:/usr/java/j2sdk1.3.1_08/bin
   ```

3. Verify that your environment variables are properly set. IDS variables should be pointing to the correct server instance, PATH should include Informix and Java bin directories, and CLASSPATH should include the Informix JDBC jar files and the current directory. In our case we included all these settings in our $INFORMIXDIR/demo/server/profile_settings file. This is how it looked:

   ```
   INFORMIXSERVER=demo_on

   INFORMIXDIR=/home/IDS9.40

   ONCONFIG=onconfig.demo

   CLASSPATH=/home/JDBC/lib/ifxjdbc.jar:/home/JDBC/lib/ifxjdbcx.jar:.

   PATH=/home/IDS9.40/bin:/home/IDS9.40/bin:/home/IDS9.40/bin:/usr/local/sbin:/usr
   /local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin:/usr/java/j2s
   dk1.3.1_08/bin:/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin:/home/informix/bin:/
   home/informix/bin:/root/bin:/usr/java/j2sdk1.3.1_08/bin:/opt/netscape/netscape:
   .

   export INFORMIXSERVER INFORMIXDIR ONCONFIG PATH CLASSPATH
   ```

4. After setting these variables you should be able to run IDS commands (as long as you have the necessary permissions), compile and run normal Java programs, and also compile and run JAVA programs to access IDS using the Informix JDBC driver.

Compiling and running the demo application

Now you are ready to compile and run the application. This program is a modified version of the SimpleConnection demo program that comes with the JDBC driver. You can find the SimpleConnection demo, and others, in the <JDBC_HOME>/demo directory. Example 2-17 shows the source code of our program.

Example 2-17 Source code of the VerifyIDSConfigForWAS program

```
import java.sql.*;
import java.util.*;

public class VerifyIDSConfigForWAS {

  public static void main(String[] args)
  {
    String url =
"jdbc:informix-sqli://neon:1533/stores_demo:informixserver=demo_on;user=itso;password=itso"
;
    String testName = "Simple Connection";
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;
    int  i = 0;

    System.out.println(">>>" + testName + " test.");
    System.out.println("URL = \"" + url + "\"");

    try
    {
      Class.forName("com.informix.jdbc.IfxDriver");
    }
    catch (Exception e)
    {
      System.out.println("FAILED: failed to load Informix JDBC driver.");
    }

    try
    {
      conn = DriverManager.getConnection(url);
    }
    catch (SQLException e)
    {
      System.out.println("FAILED: failed to connect!");
    }

    try
    {
      stmt = conn.createStatement();
      rs = stmt.executeQuery("select * from customer");

      System.out.println("");
System.out.println("****************** Customer Information ***********************");
      System.out.println("");
      System.out.print("Customer ID");
      System.out.print("     First Name");
      System.out.print("     Last Name");
      System.out.println("     Company");

System.out.println("===============================================================");
      System.out.println("");
      while(rs.next() && i < 4)
      {
        String custID = rs.getString("customer_num");
        String fName = rs.getString("fname");
        String lName = rs.getString("lname");
        String company = rs.getString("company");
        System.out.print("     " + custID);
        System.out.print("        " + fName);
        System.out.print(lName);
        System.out.println(company);
        i++;
      };
    }
    catch (Exception e)
    {
      System.out.println("Failed to run query!");
    }

    try
    {
      conn.close();
    }
    catch (SQLException e)
    {
      System.out.println("FAILED: failed to close the connection!");
    }
    System.out.println("");
    System.out.println(">>>End of " + testName + " test.");
  }
}
```

The most important information here that might need to be changed is the URL. Notice that in our case it is set to:

```
jdbc:informix-sqli://neon:1533/stores_demo:informixserver=demo_on;user=itso;password=itso
```

where **neon** is the name of our host machine, **1533** is our TCP port number to connect to IDS, **stores_demo** is our database name, **demo_on** is our server instance name (INFORMIXSERVER), and **itso** is our user.

To compile and run the program follow these steps (on the same terminal session that has the environment variables set) :

1. Copy the source code into your working directory and name it: VerifyIDSConfigForWAS.java

Attention: Don’t forget to change the connection URL information to the values used in your configuration if it is different from our values - before you go to step 2!

2. Compile the program:

   ```
   [itso@neon itso]$ javac VerifyIDSConfigForWAS.java
   ```

3. Run the program:

   ```
   [itso@neon itso]$ java VerifyIDSConfigForWAS
   ```

Example 2-18 shows the expected results.

Example 2-18   Simple connection test

```
>>>Simple connection test.
URL =
"jdbc:informix-sqli://neon:1533/stores_demo:informixserver=demo_on;user=itso;password=itso"

****************** Customer Information ***********************************
Customer ID     First Name     Last Name     Company
==================================================
     101        Ludwig         Pauli          All Sports Supplies
     102        Carole         Sadler         Sports Spot
     103        Philip         Currie         Phil's Sports
     104        Anthony        Higgins        Play Ball!
>>>End of Simple Connection test.
```

Receiving the above query results from the stores_demo database verifies that IDS is installed properly, that you can connect to it using JDBC, that you can access it with a Java application, and that it is ready to be integrated with WebSphere.

The following chapters will overview WebSphere, step you through the installation and configuration of WebSphere, and guide you through the integration of IDS with WebSphere. You will actually be stepped through the development of an application, using WebSphere Studio Application Developer, that validates the integration of IDS with WebSphere.

3

WebSphere V5: An overview

WebSphere is IBM's strategic software platform for e-business. As such, WebSphere represents a number of product packages that address all aspects of e-business. The WebSphere family represent a powerful and flexible application development and deployment environment. For a complete overview of all WebSphere product packages, please see the WebSphere product pages on the IBM Web site (`http://www.ibm.com`®).

The objective of this redbook is to demonstrate how WebSphere can be used in an integrated environment with Informix Dynamic Server (IDS). The focus of this section will be on those WebSphere product packages that were implemented in the development of this redbook, to meet that objective. The WebSphere product packages are grouped into the following three categories:

- Business Integration
- Reach and User Experience
- Foundation and Tools

In this section we focus primarily on the category of Foundation and Tools. We provide a brief overview of this selected set of the product packages that includes the basic architecture and some of the functions and features. This is not intended to be an exhaustive overview, but rather, more of an introduction for those not already familiar with WebSphere. There are more detailed descriptions of the implementation process and the use of those selected product packages in subsequent chapters of this redbook.

3.1 What is WebSphere?

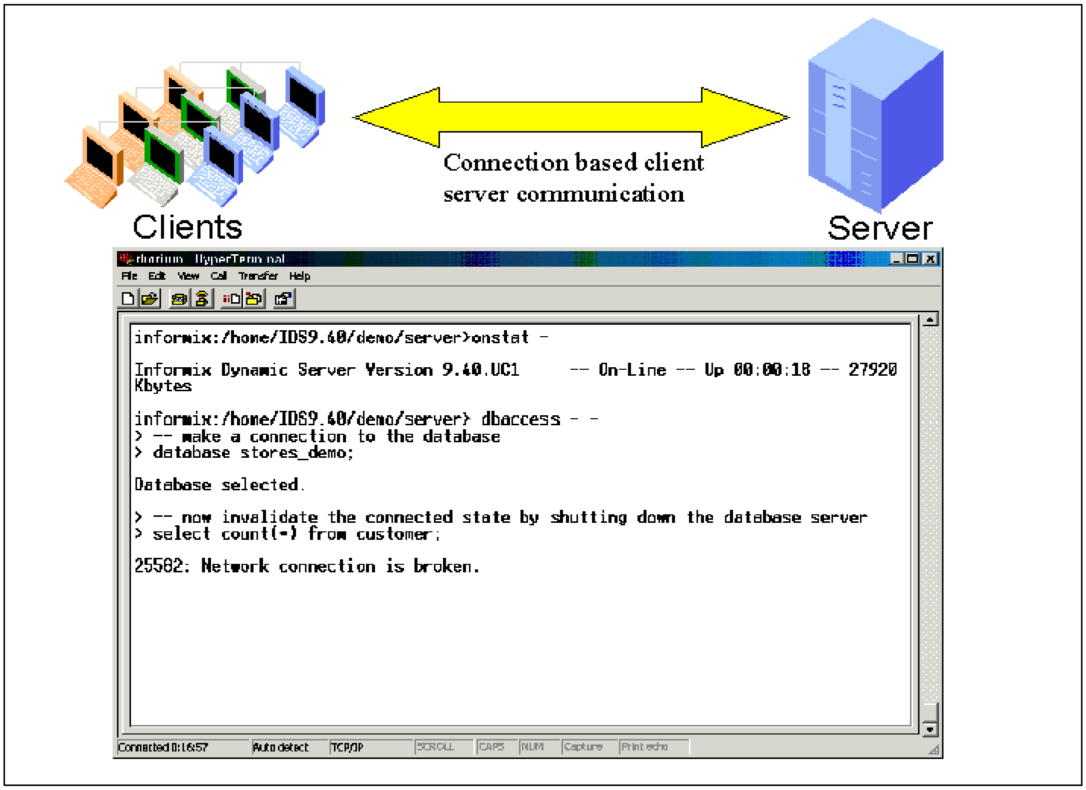
This section provides a brief introduction to WebSphere and discusses why it is important in today's fast moving, online, and Internet ready environment.

WebSphere is IBM's infrastructure software for dynamic e-business. It has evolved from its start as a Web application server, to a full set of related products and offerings with a common base. Founded on open standards, the WebSphere family of product packages is now the most comprehensive and fastest growing e-business platform on the market.

One of the main components of this e-business infrastructure is the WebSphere Enterprise Application Server. To understand what this really means we have to examine what problem WebSphere is designed to solve, and then the definition Enterprise Application Server will make much more sense. To understand the problem, we need to take a brief tour of the history of client-server architecture.

Early client server computing history: connection based

A very long time ago, dating as far back as the late 1980's and 1990's, two-tier client server computing was the defacto standard for open systems. The architecture looked as shown in Figure 3-1.



```
informix:/home/IDS9.40/demo/server>onstat -

Informix Dynamic Server Version 9.40.UC1     -- On-Line -- Up 00:00:18 -- 27920
Kbytes

informix:/home/IDS9.40/demo/server> dbaccess - -
> -- make a connection to the database
> database stores_demo;

Database selected.

> -- now invalidate the connected state by shutting down the database server
> select count(*) from customer;

25582: Network connection is broken.
```

Figure 3-1 Client-server computing model

The communication between the client computer on the left of the illustration and the server on the right was *connection* based. You are already familiar with this concept; the screen capture is designed to help clarify the definition.

When the dbaccess session shown executes the statement:

```
>database stores_demo
```

it is requesting a connection to the stores_demo database.

The following message indicates that this was successful:

```
Database selected.
```

A connection has been established, and the session can be viewed with **onstat –g ses**.

We can demonstrate a connection failure by forcing the server to stop by using **onmode -yuk**.

The following message reminds us that the active communication between the client and server has been violated, and that a session between the client and server no longer exists:

```
25582: Network connection is broken.
```

Connection oriented computing made client server computing straightforward. But in the mid-1990s, the World Wide Web thrust connectionless computing into center stage, and the relatively simple days of two-tier client server computing would never be the same.

The World Wide Web: Connectionless Computing

The original design of the World Wide Web was to publish documents for easy browsing. Connection computing was rejected, because a typical server didn't have to resources to track hundreds, if not thousands, of sessions over what was often a slow or unreliable network.

By eliminating the connection, and need to track the state of the clients connecting to the server, a much smaller server could handle many clients on the net, as shown in Figure 3-2.

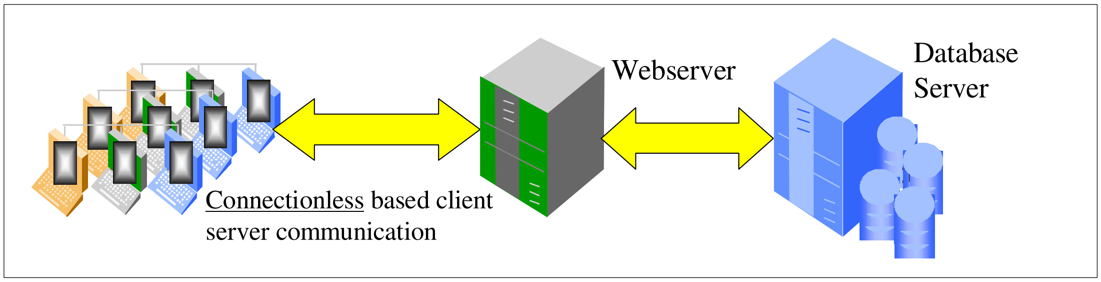


Figure 3-2 Connectionless computing

A simple text markup language, HyperText Markup Language (HTML), was not sufficient to provide access to important data sources such as databases, so the Communications Gateway Interface (CGI) was created. CGI allows Web servers to run programs at the operating system level of the Web server. In very short order, CGI was mated with Perl and other languages in order to get access to databases.

The issue of connections and sessions still had to be addressed, and this was remedied by the use of *cookies*. A cookie is a unique identifier for the client that is sent by the browser back to the Web server with each page request.

This is not a good replacement for the robust two-tier client server model. Then, to add even more complexity, a new language called Java became extremely popular for Web development.

Early Java computing

The Java programming language added great flexibility and function to the Web environment, and also cemented the need for application server technology, as shown in Figure 3-3.

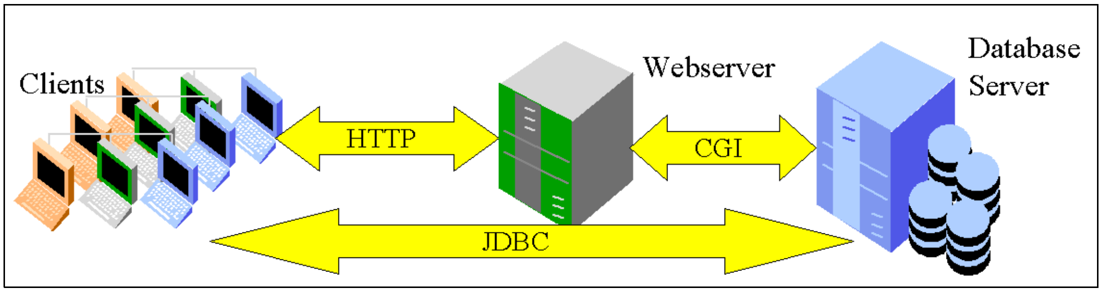


Figure 3-3 Early Java Computing

The initial Java design used a command line run time Java Virtual Machine (JVM) to launch applets or a run time environment hosted by a browser.

The scalability issues are obvious. For example, the application can't be very large if the run time is resident in the client browser. And, the applications had to be downloaded to the browser which introduced network bandwidth issues. Database connections used either CGI or Java Database Connectivity (JDBC). With a direct JDBC connection between the client and the database, connection pooling couldn't be leveraged.

Other issues were spawned by the *browser wars* due to incompatibilities between JVMs in different browsers. The security model of the JVM dictated no local access to the surrounding operating system, again limiting the power of the applications.

WebSphere to the rescue

WebSphere is specifically designed to address the issues raised in our brief survey of Web client-server computing, as shown in Figure 3-4.

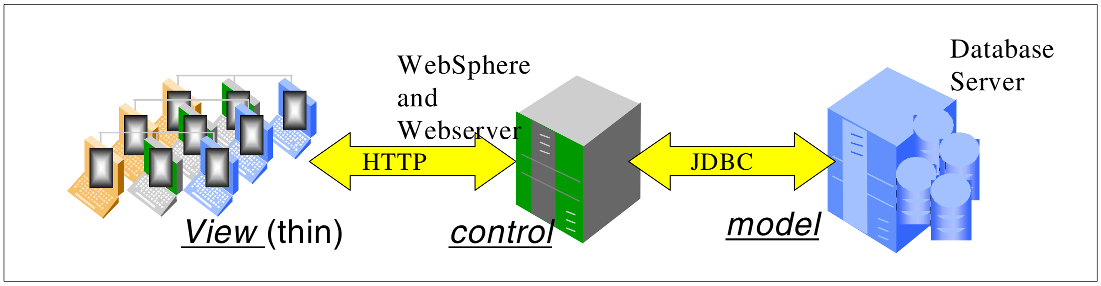


Figure 3-4 Application model

Here is a simple definition of WebSphere:

WebSphere is an industrial strength home for your Java applications.

WebSphere addresses the session persistence issues, JVM issues, and database access and pooling, all while conforming to open industry Java 2 Enterprise Edition (J2EE) Java architecture (for more details, see `http://www.javasoft.com`).

The client session state is maintained by cookies, and WebSphere uses robust database technology to make the session information persistent. This provides scalability and fault tolerance as many servers can access the database containing session information.

WebSphere application development is in industry standard environments including Java Server Pages (JSP) and Enterprise Java Beans (EJB). The MVC – Model, View, Control architecture significantly improves programmer productivity. The MVC architecture will be explored in greater detail in “Application model” on page 69.

WebSphere runs as a middle tier server. It can be hosted on anything from a laptop to a high end server. Vertical and horizontal scaling provides even greater scalability. This solves the issues associated with running large Java applications on the client desktop. The middle tier also provides database connection pooling, so the overhead of connecting to the database is eliminated.

So, WebSphere has become the IBM infrastructure for dynamic e-business. It has evolved from its start as a Web application server, to a full set of related products and offerings with a common base. Founded on open standards, the WebSphere family of product packages is now the most comprehensive and fastest growing e-business platform on the market.

Now let’s take a brief look at some of the WebSphere family of products.

3.2 WebSphere product family

Designed as a pyramid, the WebSphere platform model illustrates the depth and breadth of the WebSphere product family. The product packages that comprise the WebSphere family are categorized in three groups, as depicted in Figure 3-5.

- **Foundation and tools:** These products represent the functional infrastructure. Included are WebSphere Application Server, WebSphere Studio, and WebSphere Host Integration. The Application Server is the runtime environment of choice. It represents an enterprise-level runtime environment and provides workload management, scalability, and security. WebSphere Studio provides the development environment. It has a range of tools and functions for the development and testing of Web based applications.
- **Reach and user experience:** These products extend range of the applications and enable them to reach new customers. For example, they can help a company move their business from traditional channels to the internet, and enable the creation of online stores. Included are WebSphere Portal, WebSphere Everyplace®, and WebSphere Commerce.
- **Business integration:** The products in this category are WebSphere Business Integration, and WebSphere MQ Integrator Broker. They help customers enable interoperability by interconnecting and integrating the various sources of data and information. WebSphere MQ provides an underlying message-oriented infrastructure, Java Messaging Services (JMS) support, a channel for Simple Object Access Protocol (SOAP) message exchange, and other options such as HTTP and HTTPS.

WebSphere is a robust, comprehensive family of product packages. The focus of this RedBook is WebSphere Application Server (WAS) and WebSphere Studio Application Developer (WSAD). However, we do provide some discussion of other WebSphere products in this redbook, such as Portal Server and WebSphere MQ.

In addition, redbooks are available that provide more detail on all products in the WebSphere family. For information on other WebSphere products, go to:

```
http://www.redbooks.ibm.com
```

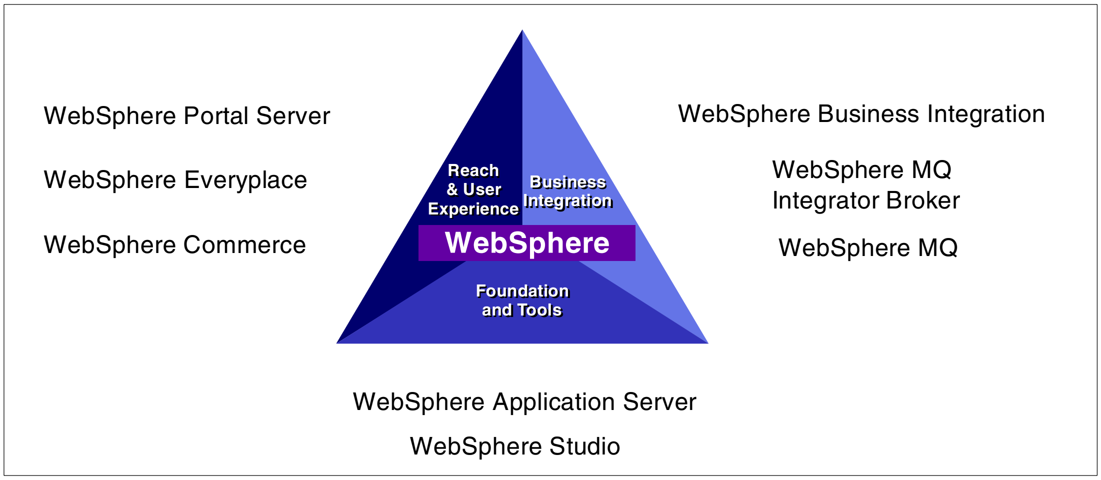


Figure 3-5 WebSphere platform model

3.3 WebSphere Application Server (WAS)

WebSphere Application Server is a key element of a business-on-demand infrastructure. Version 5 provides an integration of the deployment model, administration point, programming model, and integrated application development environment, and is fully J2EE 1.3 compliant. It is available in multiple packages to satisfy a wide range of user requirements. The range is from a simple single server to a clustered, highly available, high-volume environment. The product packages are also available for a range of platforms, including Windows NT/2000/XP, Linux, AIX®, Sun Solaris, HP-UX, and z/OS®.

In this section we present some of the J2EE concepts and model, the system architecture of WebSphere Application Server and its components, and finally we show a table comparison of the different application server editions that are available.

3.3.1 J2EE: Overview

WebSphere is IBM's software platform for e-business and it is based on the J2EE industry standard model. In this section we present a conceptual overview of the J2EE standard and describe the architecture features of WebSphere Application Server.

What is J2EE?

The acronym J2EE stands for Java 2 Platform, Enterprise Edition. The J2EE platform is the standard for developing, deploying, and running multi-tier enterprise applications. J2EE simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many of the details of application behavior automatically, without complex programming. The standard architecture defined by J2EE is composed of the following elements:

- Standard application model for developing multi-tier applications.
- Standard platform for hosting applications.
- Compatibility test suite for verifying that J2EE platform products comply with the J2EE platform standard.
- Reference implementation providing an operational definition of the J2EE platform.

The J2EE platform specification describes the runtime environment for a J2EE application. This environment includes application components, containers, and resource manager drivers. The elements of this environment communicate with a set of standard services that are also specified.

J2EE benefits

The J2EE standard empowers customers, so they can compare J2EE offerings from vendors and know that they are comparing apples with apples. Comprehensive, independent Compatibility Test Suites ensure vendor compliance with J2EE standards.

These are some benefits of deploying to a J2EE-compliant architecture:

- A simplified architecture based on standard components, services and clients, that takes advantage of the write-once, run-anywhere Java technology
- Services providing integration with existing systems, including Java DataBase Connectivity (JDBC); Java Message Service (JMS); Java Interface Definition Language (Java IDL); JavaMail; and Java Transaction API (JTA and JTS) for reliable business transactions
- Scalability to meet demand, for example, by distributing containers across multiple system and using database connection pooling
- A better choice of application development tools, and components from vendors providing off-the-shelf solutions
- A flexible security model that provides single sign-on support, integration with legacy security schemes, and a unified approach to securing application components

The J2EE specifications are the result of an industry-wide effort that has involved, and still involves, a large number of contributors.

J2EE application and programming model

Here we describe the J2EE multi-tier application (system) model, explaining the different tiers and what software is typically installed on each of them. We also present the J2EE programming model, as well as their parts and how they relate to each other.

Application model

The J2EE application model is comprised by a n-tier environment where a number of tiers of application logic and business services are separated into components that communicate with each other across a network. The basic form is a logical three-tier computing model, which we depict in Figure 3-6.

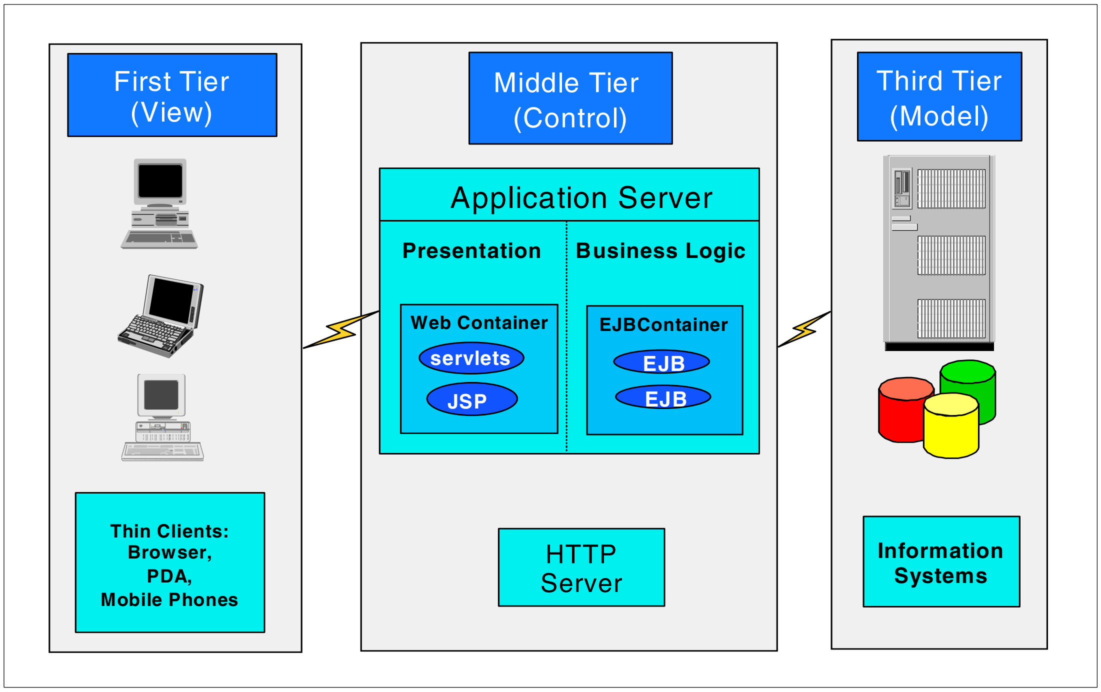


Figure 3-6 J2EE Application model

Note: These are logical tiers, so it does not mean that they necessarily are running on three different physical servers.

The three tiers can be described as follows:

- **First Tier (View):** This tier consists of thin clients. Their main function is to present information and results, produced by an application, to the user. They are called thin clients because little or no application logic is executed on the client, hence relatively little software is required to be installed on the client. The common element that ties these clients to the Web application server is their implementation of a set of widely supported Internet-based technologies and protocols, along with Java, which enables them to provide interaction between users and applications. Examples of thin clients are Web browsers (Netscape or Internet Explorer), PDA's, and mobile phones.
- **Middle Tier (Control):** This tier is typically comprised of a standard-based Web server (for example, IBM HTTP Server) that interacts with the client tier and defines user interaction. The Web application server (for example, WebSphere Application Server) executes business logic independently of the client type and user interface style. It is implemented using various Internet and Java technologies, including the HyperText Transfer Protocol (HTTP) server and the Enterprise Java services that enable rapid development and deployment of applications in a distributed network environment.

  Java Servlets, Java Server Pages, and Enterprise Java Beans are examples of the components deployed in the Web application server. These server-side components communicate with their clients and other application components via HTTP or IIOP, and use the directory and security services provided by the network infrastructure. They can also leverage database, transaction, and groupware, facilities. This middle tier can also be logically divided into the presentation tier and the business logic tier. Typically Enterprise Java Beans process the business logic, and JSPs and Servlets are responsible for presenting the results to a client request.

- **Third Tier (Model):** This tier primarily consists of the Customer Information Control System (CICS®) server, legacy applications developed on mainframes, and legacy systems, or relational databases such as IDS.

Programming model

The J2EE programming model (Figure 3-7) categorizes enterprise applications into components, containers, and connectors, as follows:

- **Components:** These are the key focus of application developers, whereas system vendors implement containers and connectors to conceal complexity and promote portability. Typical J2EE components include: Servlets and Java Server Pages (JSP), Enterprise Java Beans (EJB), application clients, and Applets.
- **Containers:** These intercede between clients and components by providing services transparently to both, including transaction support and resource pooling. Container mediation allows many component behaviors to be specified at deployment time, rather than in program code. Containers are provided by application server vendors, an example being IBM WebSphere application server. An application server provides containers for components listed above, so Enterprise Java Beans are executed by the EJB container, servlets and JSP are executed by the Web container, application clients are executed by the application client container, and finally applets are executed by the applet container.
- **Connectors:** These sit beneath the J2EE platform, defining a portable service API to plug into existing enterprise vendor offerings. Connectors promote flexibility by enabling a variety of implementations of specific services. Connectors are usually implemented by database vendors, such as IBM provides with Informix Dynamic Server.

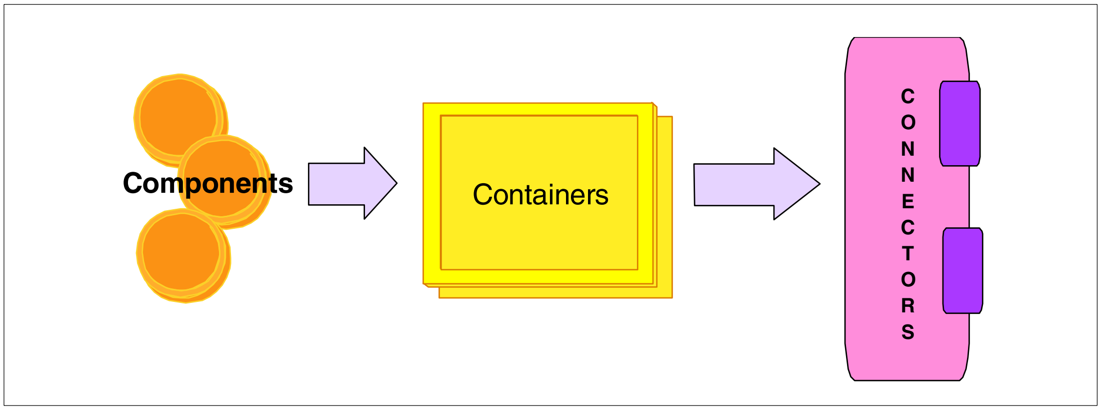


Figure 3-7 J2EE programming model

3.3.2 WebSphere Application Server: Architecture

IBM WebSphere Application Server, Version 5 has completed the full J2EE certification test suite. The product supports all of the J2EE 1.3 APIs, and exceeds many with its extensions. In this section we look at the architecture of WebSphere Application Server and describe its major components, including the application server itself, the Web container, and the EJB container, and describe how they map to the J2EE model. See Figure 3-8.

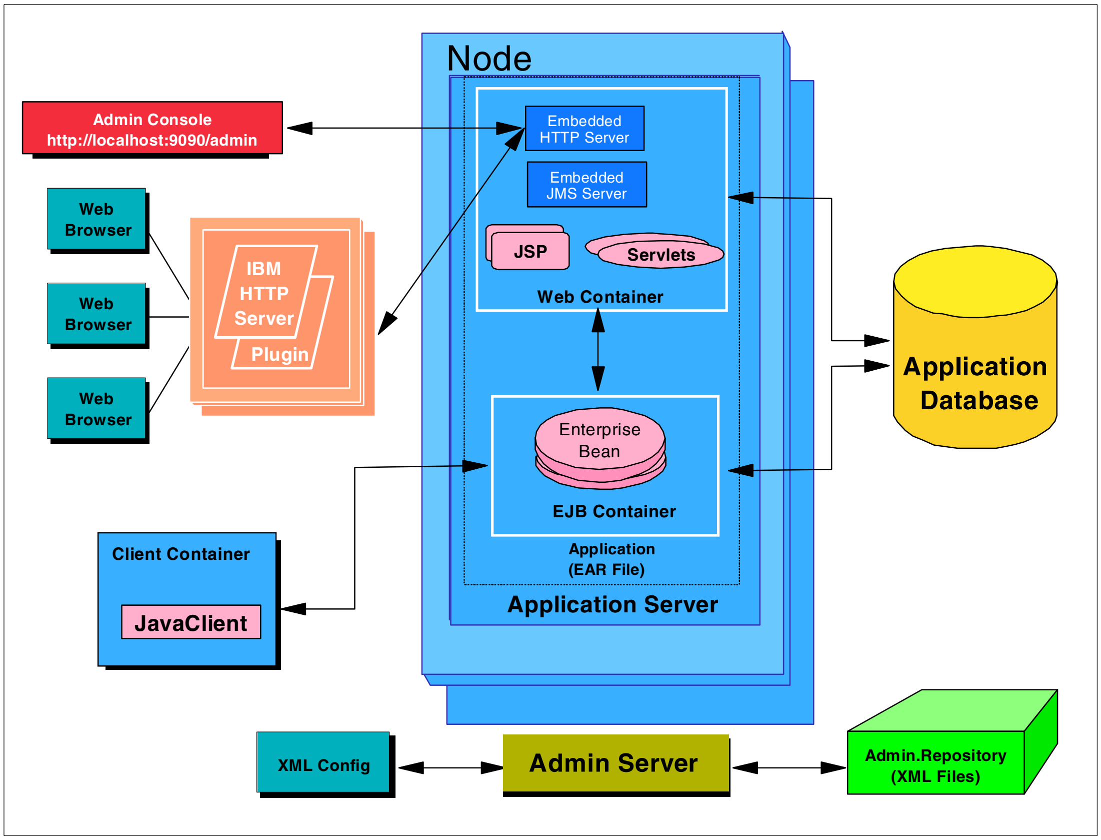


Figure 3-8 WAS architecture

Node

A node is the physical machine (server) where the application server is installed.

Application server

The application server collaborates with the Web server to return customized response to a client request. Application code, including servlets, JSPs, EJBs, and their supporting classes, run in an application server. In keeping with the J2EE component architecture, servlets and JSPs run in a Web container, and EJBs run in an EJB container. You can define multiple application servers, each running in its own Java Virtual Machine (JVM).

- **Web container:** Servlets and JavaServer Pages (JSP) are server-side components used to process requests from HTTP clients, such as Web browsers. They handle presentation and control of the user interaction with the underlying application data and business logic. They can also generate formatted data, such as XML, for use by other application components. The Web container processes servlets and JSP files. 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 destroy method when appropriate and unloads the servlet, after which the JVM performs garbage collection.
- **EJB container:** The EJB container provides the runtime services needed to deploy and manage EJB components, known as enterprise beans. It is a server process that handles requests for session, entity, and message-driven beans. The enterprise beans (inside EJB modules) installed in an application server do not communicate directly with the server; instead, an EJB container provides an interface between the enterprise beans 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 hold more than one EJB jar file.

HTTP server

The WebSphere Application Server works with an HTTP server, or Web server, to handle requests for dynamic content, such as servlets from Web applications. The HTTP server and application server communicate using the WebSphere HTTP plug-in for the HTTP server. The HTTP plug-in uses an easy-to-read XML configuration file to determine whether a request should be handled by the Web server or the application server. It uses the standard HTTP protocol to communicate with the application server. It can also be configured to use secure HTTPS, if required. The HTTP plug-in is available for popular Web servers.

Application client container

Application clients are Java programs that typically run on a desktop computer with a graphical user interface (GUI). They have access to the full range of J2EE server-side components and services. The application client container handles Java application programs that access enterprise beans, Java Database Connectivity (JDBC), and Java Message Service message queues. The J2EE application client program runs on client machines. This program follows the same Java programming model as other Java programs; however, the J2EE application client depends on the application client run time to configure its execution environment, and uses the Java Naming and Directory Interface (JNDI) namespace to access resources.

Embedded JMS server

The embedded JMS server provides built-in support for Java Message Service (JMS) messaging between WebSphere applications, including message-driven beans.

Embedded HTTP server

The embedded HTTP server enables HTTP clients to connect directly to the application server. Or, as previously described, an HTTP client can connect to a Web server and the HTTP plug-in can forward the request to the application server.

3.3.3 WebSphere Application Server: Packages

Here we provide brief descriptions of the WebSphere Application Server product packages that are currently available (a high level function/feature comparison of the different packages is given in Table 3-1 on page 74):

- **WebSphere Application Server - Express** offers an affordable solution for managing simple dynamic Web sites. It is Java-based and supports a development environment using WebSphere Studio. With it you can convert static Web sites into dynamic Web applications. It is an entry-level configuration, but has a migration path to other WebSphere Application Server packages for growth and investment protection.
- **WebSphere Application Server** is also a Java-based application server, with integrated enterprise data and transaction support for the e-business world. It offers a set of application services to enable and manage heterogeneous Web services, increased security, performance, availability, connectivity, and scalability. It supports standards such as XML, SOAP, and Web services description language (WSDL). It also extends the Java 2 Enterprise Edition (J2EE) 1.3 programming model by providing an infrastructure to support production-ready deployment of Web services-based applications. This makes it possible to deploy Web services with the communication mechanism of your choice, including SOAP and HTTP, Java Message Service (JMS), or Remote Method Invocation Internet Inter-ORB Protocol (RMI-IIOP).

- **WebSphere Application Server Network Deployment** provides advanced Web services and clustering capabilities. It offers a private UDDI registry for easier deployment of internal Web services applications and the Web Services Gateway for external applications that request Web services from an internal Web services provider application.
- **WebSphere Application Server Enterprise Edition** provides an advanced application server configuration that simplifies build-to-integrate tasks, accelerates application development, and enables dynamic application flexibility. It offers build-to-integrate capabilities such as advanced application adapters, application workflow composition and choreography, extended messaging, dynamic rules-based application adaptability and internationalization, and asynchronous processing. It delivers more integration capabilities than WebSphere Application Server Network Deployment through extensions beyond the specifications of the J2EE standards

WebSphere Application Server has many benefits, among them are:

- Full J2EE 1.3 compatibility, modular, and standards based, means easier and less expensive application development.
- Seamless configurations and administration, based on Java Management Extensions (JMX), with a browser-based administration facility.
- Improved programmer productivity and simplified enterprise development with JMS API. It includes standards, such as XML, SOAP, and WSDL, as well as enhanced options, such as the Web Services Gateway, and a private UDDI registry.
- Enhanced security model supports open standards-based Java specifications, such as Java 2 Security, JAAS, and WebSphere's pluggable security architecture.

For more information on IBM WebSphere Application Server, refer to:
`http://www-3.ibm.com/software/webservers/`

WebSphere Server V5.x: Product package comparisons

Table 3-1 provides a comparison of the various product packages.

Table 3-1 Product comparisons for WebSphere Server

Description	**Exp**	**AS**	**ND**	**Ent**
Java Programming Model				
Support for JavaServer Pages 1.2 and Java Servlets 2.3	x	x	x	x
Full J2EE 1.3 Support		x	x	x
Support for some features planned for J2EE 1.4[1]			x	x
Full XML Support	x	x	x	x
Web services				
Full Web services support	x	x	x	x
Support for private UDDI registries			x	x
Web Services Gateway			x	x
Database support and Connectivity				
JDBC Connection Management for Informix	x[2]	x[2]	x[2]	x[2]
Application Development				
Sample Applications	x[3]	x	x	x
Development tool based on WebSphere Studio	x			
Integrated J2EE Technology-based workflow engine				x
Business rule beans				x
Business process beans				x

Description	**Exp**	**AS**	**ND**	**Ent**
Asynchronous processing	x	x	x	x
Internalization Service				x
Web server support				
Embedded HTTP Server	x	x	x	x
IBM HTTP Server included[4]		x	x	x
Web server plug-ins	x[5]	x	x	x
Performance support				
Enhanced features for performance such as dynamic caching, IBM Tivoli Performance Viewer, integration with third-party tools		x	x	x
CORBA C++ SDK				x
Security				
Basic authentication and authorization for secure access to Web Resources	x	x	x	x
Enhanced authentication and authorization through Common Secure Interoperability (CSI), Version 2.0, single sign-on and support for LDAP	x[5]	x	x	x
Advanced Authentication and Authorization, such as Java authentication authorization service (JAAS) and Java Cryptographic Extension (JCE) for enhanced Security	x[5]	x	x	x
Convenient security administration through embedded admin console	x[5]	x	x	x
Application Connectivity				
Full Java Message Service (JMS) supported message driven beans, including embedded JMS transport		x	x	x
Microsoft Common object model architecture to EJB support for integration with ActiveX client and server resources		x	x	x
Restricted WebSphere MQ license included				x
Advanced transaction support				x
Application profiling				x
Dynamic and extended query capabilities				x
Container managed messaging				x
Administration and workload management				
Simplified administration using the development tool interface	x			
Browser-based administration for remote administration across firewalls	x[5]	x	x	x
Intelligent workload distribution across a cluster			x	x
Failure bypass			x	x
Clustering support			x	x
Migration Support				
Migration Documentation	x[5]	x	x	x
Migration tools and assistance	x[5]	x	x	x

COLUMN HEADING DEFINITIONS:
EXP = WebSphere Application Server -Express 5.x
AS = WebSphere Application Server - 5.x
ND = WebSphere Application Server -Network Deployment 5.x
ENT = WebSphere Application Server -Enterprise 5.x

VALUE NOTES (Superscripts):
1 - Features which are currently planned for J2EE 1.4 Spec.
2 - Not supported for iSeries™
3 - Samples are integrated with the tool for quick-start development
4 - IBM HTTP Server included with IBM OS/400®
5 - Available on WebSphere application Server - Express for iSeries, V5.0 only

IBM WebSphere Application Server, Version 5

In this section we list the hardware requirements for supported operating systems.

Disk and hardware requirements — all systems

- Minimum of 200 MB available disk space for installation (including IBM Software Developer Kit)
- CD-ROM Drive
- Support for a communications adapter

Recommended memory requirements all systems

- Minimum of 256 Mb memory, 512Mb recommended

Operating system and processor requirements

In this section we cover the requirements for the operating system and processor.

Windows NT® Windows 2000

- An Intel technology-based PC running Microsoft Windows NT Server, Version 4.0 Service Pack 6a or higher; or Windows 2000 Server or Windows 2000 Advanced Server with Service Pack 3 or higher
- Intel Pentium processor at 500Mhz or faster

AIX

- IBM RS/6000® or RS6000 SP running IBM AIX, Version 4.3.3 with 4330-10 recommended maintenance package, or Version 5.1 with the 5100-02 recommended maintenance package
- IBM RS/6000 604e workstation at 375Mhz or faster

Linux operating environment on Intel

- Red Hat Linux Advanced Server 2.1, Red Hat Linux Server 7.2 or 8.1 SuSE 7.3, or SLES 7.0, based on kernel 2.4
- Intel x86 processor at 500Mhz or faster
- Support for TCP/IP and an appropriate communications adapter
- Legacy application support RPMs installed

Red Hat Linux on IBM eServer™ zSeries® operating environment

- zSeries with Linux distribution SuSE 7.0 or Red Hat Linux 7.2 or 8.1, based on kernel 2.4
- G5, G6, or higher processor
- Legacy application support RPMs installed

Sun Solaris operating environment

- A workstation running Sun Solaris operating environment, Version 8 at maintenance level of July 2002 or higher
- Sparc workstation at 440Mhz or higher

3.4 WebSphere Studio Application Developer (WSAD)

In this section we focus on Web application development. WebSphere Studio Application Developer is a set of development tools for enterprise e-business Java-based applications. WebSphere Studio can be tailored to meet the specific development needs of your growing business — from simple Web site creation to complex e-commerce and enterprise application development.

We begin the section with the background of Eclipse, since WebSphere Studio is a commercial implementation of the Eclipse project. Then we show the role-oriented development model of the Eclipse/WebSphere Studio workbench and the WebSphere Studio packages that are available. Finally we list some of the benefits of using Studio as a development environment, and briefly discuss the development roadmap for Informix 4GL and its evolution towards integration with WebSphere Studio.

Eclipse

One of the primary challenges for a software developer is to integrate existing tools that are supplied from different vendors. Typically tools from different vendors do not work together or, if they do, the task of making them interact smoothly with each other can be very time consuming and resource intensive.

One proposed solution for this problem is an open and extensible tool platform that provides improved developer productivity, reliability, extensions through plug-ins, and a broad array of tool providers. Having this vision in mind a new open source community was created, called eclipse.org. Industry leaders like IBM, Borland, MERANT, QNX Software Systems, Rational® Software, Red Hat, SuSE, TogetherSoft, and Webgain2 formed the initial eclipse.org Board of Stewards in November 2001. Since then many other companies have joined the Board of Stewards and all of them have made a commitment to releasing Eclipse Platform compatible product offerings to the community of users, researchers, and developers.

The Eclipse platform defines the set of frameworks and common services that collectively make up *integration-ware* required to support a comprehensive tool integration platform. These services and frameworks represent the common facilities required by most tool builders including a standard workbench user interface and project model for managing resources, portable native widget and user interface libraries, automatic resource delta management for incremental compilers and builders, language-independent debug infrastructure, and infrastructure for distributed multi-user versioned resource management.

WebSphere Studio Application Developer is one of the WebSphere Studio family of products that has been developed based on the Eclipse Workbench. The WebSphere and Eclipse Workbench is depicted in Figure 3-9.

.

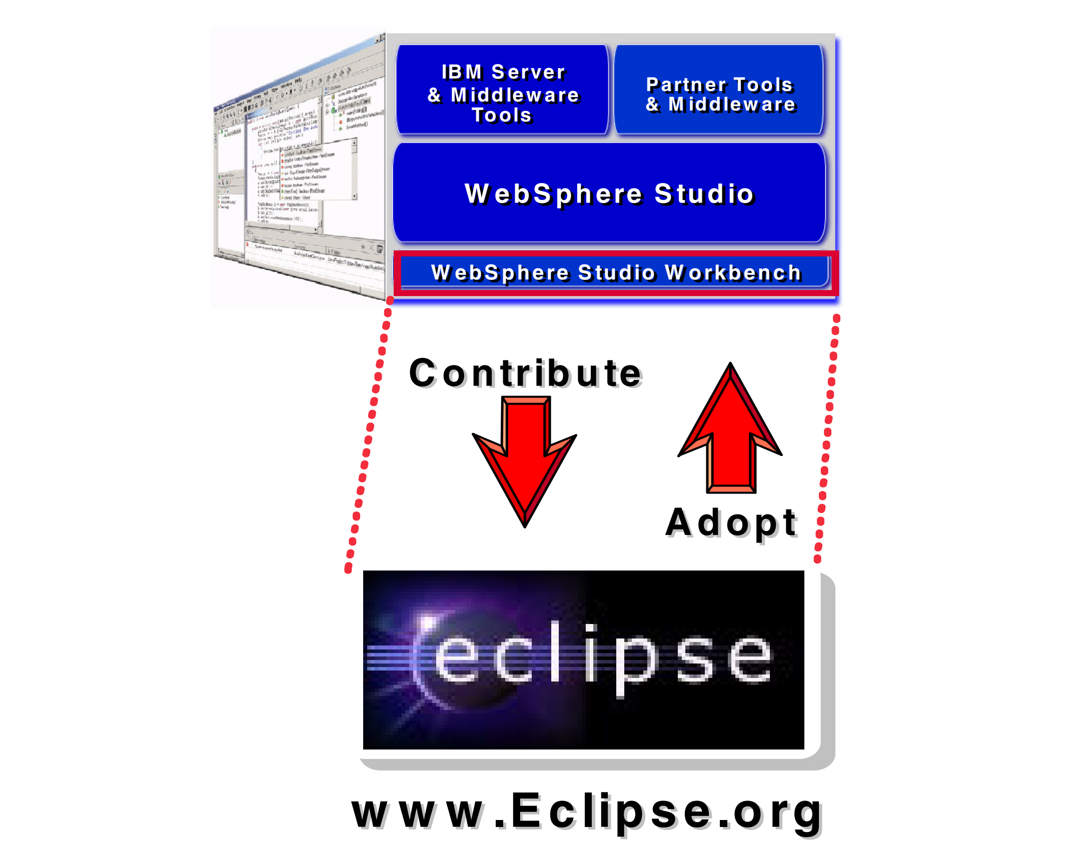

IBM Server & Middleware Tools
Partner Tools & Middleware
WebSphere Studio
WebSphere Studio Workbench
Contribute
Adopt
eclipse
www.Eclipse.org

Figure 3-9 WebSphere and Eclipse workbench

3.4.1 Role-based development model

In essence, the workbench provides an integrated development environment (IDE) that has flexible perspectives. It is designed to provide special support for a particular e-business development role, or for a range of roles. These perspectives contain views and editors that provide a common way for members of the development team to create, manage, and navigate resources. For example, a Java developer would work most often in the Java perspective, while a Web designer would work in the Web perspective.

Within the workbench based products, task-oriented perspectives filter out much of the overall Web and Java development complexity, and present the developer only with those functions that are relevant to the task at hand. Users can switch perspectives depending on what they are working on at any given moment, or depending on their current role in the project. Because different developers are accustomed to working in different ways, any perspective can be further customized. Since they are built using the Eclipse Workbench technology, all tools and perspectives share a common look and feel, which reduces learning curves and helps maximize developer productivity.

Most of the perspectives use a similar layout, depicted in Figure 3-10.

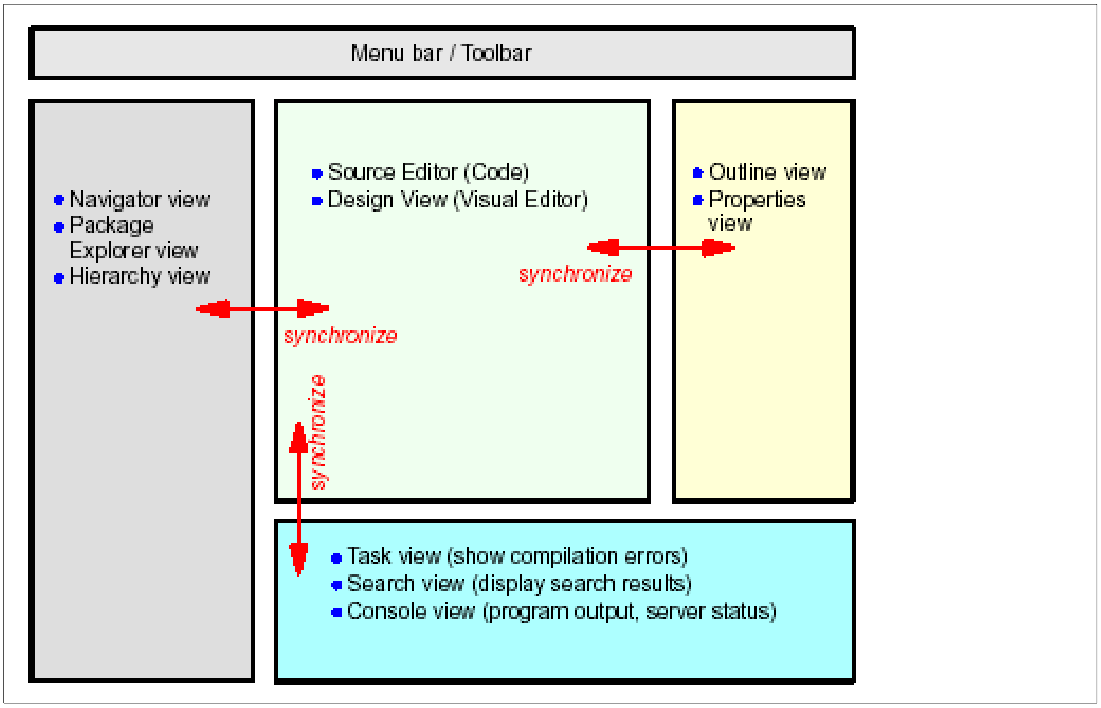


Figure 3-10 Generic perspective layout

The left side of the above figure depicts views that help you to navigate through your project's files, while in the center of the workbench you find a larger frame, usually for the source editor or the design frame. This allows you to change the code and design files in your project. The right side of the figure depicts the Outline or the Properties views. In some perspectives you can see that the editor frame is a little larger and the Outline or Properties view is placed at the bottom left corner of the perspective. The content of the views is synchronized. This means if you would change, for example, a value in the Properties view, the Editor view is automatically updated to reflect the change. These are the primary perspectives:

- **J2EE Perspective:** The J2EE Perspective provides views that you would typically use when you develop resources for enterprise application, EJB, Web, and application client or connector projects or modules.
- **Resource Perspective:** The Resource Perspective is a very simple perspective that shows the resources that are present in the workbench.
- **Web Perspective:** The Web Perspective contains many views and is used by Web developers. They can use the Web perspective to build and edit Web resources, such as servlets, JSPs, HTML pages, Style sheets, and images, as well as the deployment descriptor file, web.xml.
- **Java Perspective:** The Java Perspective supports developers who create, edit, and build Java code.
- **Server Perspective:** The Server Perspective is used to manage the server test environments you use when testing, debugging, and profiling your applications.
- **Data Perspective:** The Data Perspective lets you access relational databases tools, and you can create and manipulate the data definitions for your project. This perspective also lets you browse or import database schemas in the DB Servers view, create and work with database schemas in the Data Definition view, and change database schemas in the table editor. You can also export data definitions to another database installed either locally or remotely.

- **XML Perspective:** The XML Perspective contains several editors and views that can help a developer when building XML files, XML schemas, DTDs, stylesheets, and integrating data extracted from relational databases and XML.

For more information about the WebSphere workbench perspectives, see *WebSphere Studio Application Developer Version 5 Programming Guide* (SG24-6957-00)

3.4.2 WebSphere Studio Application Developer - Packages

The WebSphere Studio family has several member product packages. All of the packages are based on the common Eclipse Workbench, but there are also many accompanying tools and plug-ins that are tailored to the different needs of the developers. Following are descriptions of the WebSphere Studio product packages:

- **IBM WebSphere Studio Site Developer:** The Site Developer package is a set of tools and perspectives targeted for developers of Web sites and Web applications. The tools support open Web standards such as XML, JavaServer Pages, and Web services. The tools inherited from the Workbench also support Java and JavaScript development. In addition, there is a rich set of media tools required for developing a high-quality user interface. The Web services tools include those for creating services from Java components and publishing their descriptions to a UDDI registry. It is possible to browse a UDDI registry for available services and link to them from the Web application via JavaServer Pages (JSP). The Common Versions System (CVS) repository interface is a part of the Studio Workbench. In addition, Site Developer provides an interface to Rational's ClearCase® LT and includes a ClearCase LT repository. A WebSphere test environment is also part of the package.
- **IBM WebSphere Studio Application Developer:** Application Developer contains all the functionality of the Site Developer. In addition, it contains the tools for J2EE 1.2 and J2EE 1.3 development, as well as a set of profiling tools. The WebSphere test environment provides full J2EE support, including EJBs.
- **IBM WebSphere Studio Application Developer Integration Edition:** Application Developer Integrated Edition is targeted for professional developers of J2EE applications and Enterprise Extensions. It contains all the functionality of the Application Developer. In addition, it contains J2C support, Web services invocation framework, workflow and Web flow modeling tools, an enhanced WSDL editor, and an integrated WebSphere Application Server Enterprise Edition.
- **IBM WebSphere Studio Enterprise Developer:** Enterprise Developer is targeted to developers and integrators, who can make use of advanced tooling such as RAD or visual modeling. It contains all the functionality of the Application Developer. In addition, it provides tools for remote edit-compile-debug for host COBOL and PL/I applications, a visual builder for adapters and microflows, and tools for visual modeling and RAD (based on VisualAge® Generator technology).

WebSphere Studio: Product package comparisons

Table 3-2 provides a comparison of the various product packages.

Table 3-2 Product comparisons for WebSphere Studio

Description	HPB	DD	SD	AD	ED
Tools Support					
Eclipse technology-based user interface		X	X	X	X
Web Tools	X		X	X	X
Struts support			X	X	X
Relational database tools			X	X	X
XML tools			X	X	X
Web Services			X	X	X
Java tools		X	X	X	X
Team support		X	X	X	X
Integrated unit test environment			X	X	X
Profiling and tracing tools		X		X	X
J2ME tools		X			
J2EE tools				X	X
Generated EJB test client				X	
Clear case LT (included)				X	X
Java Message Service (JMS) tools					X
Java Connector Architecture (JCA) tools				IE	X
Visual workflow tools				IE	X
Service-oriented architecture				IE	X
Assembler, COBOL and PL/1 tools					X
Enterprise Generation Language (EGL) tools					X
Runtime Support					
Tomcat		X	X	X	X
WebSphere Application Server - Express			X	X	
WebSphere Application Server			X	X	X
WebSphere Application Server Enterprise				IE	X
WebSphere Micro Environment		X			
COLUMN HEADING DEFINITIONS: HPB = Homepage Builder DD = Device Developer SD = Site Developer AD = Application Devleoper ED = Enterprise Developer **VALUE NOTES:** IE - Feature only available with Integration Edition.					

WebSphere Studio Application Developer V5 became available at the end of 2002. For more information on IBM WebSphere Studio Site Developer, refer to:

`http://www.ibm.com/software/ad/studiositedev/`

For more information on IBM WebSphere Studio Application Developer, refer to:

`http://www.ibm.com/software/ad/studioappdev/`

There are many benefits of WebSphere Studio as an Application development environment. Table 3-3 provides a summary of some of the highlights of selected WebSphere Studio product packages.

Table 3-3 Summary of WebSphere Studio product packages

Product Package	Highlights
IBM WebSphere Studio Homepage Builder	Create and publish Web sites with ease with a user-friendly interface, easy-to-use wizards, templates and support for popular development languages like JavaScript, Dynamic HTML, and Cascading Style Sheets (CSS).
IBM WebSphere Studio Site Developer	Build, test, and maintain dynamic Web sites, applications, and Web services, using Java, JavaScript, and JavaServer Pages.
IBM WebSphere Studio Application Developer	Optimize and simplify J2EE application development through best practices, templates, code generation, and a comprehensive development environment with WebSphere Studio Application Developer. WebSphere Studio Application Developer Integration Edition accelerates development by adding visual workflow to build and integrate complex applications.
IBM WebSphere Studio Enterprise Developer	Give J2EE capabilities, rapid application development and team support to diverse enterprise application development organizations.
IBM WebSphere Studio Device Developer	Create and test applications that will be deployed on handsets and other mobile computing devices.
IBM WebSphere Studio Asset Analyzer	Maintain and extend existing enterprise assets through impact analysis and graphical application understanding.
IBM WebSphere Studio Application Monitor	Resolve performance problems with J2EE applications running on the IBM WebSphere for z/OS ™ platform, without requiring modification to the application code.
Toolkits for IBM WebSphere Studio	Quickly and easily add new function and tools targeted at application development needs.

Here are some of the specific benefits of WebSphere Studio:

- It helps reduce the skills needed to develop component-based Web applications.
- It lets developers rapidly create well-structured e-business systems that integrate WebSphere software and traditional transactional environments, including CICS and IMS™.
- It promotes the reuse and transformation of existing applications to reduce costs and shorten the development cycle.
- It automates the adoption of industry-standard e-business architecture through visual construction facilities based on open Struts implementation.
- It helps developers create dynamic Web applications, with support for J2EE,XML, and Web services technologies.
- It improves productivity to develop z/OS system-based applications while making the transition to e-business architecture.

- It helps transform and enable developers by facilitating skill transfer and knowledge in mission-critical enterprise technologies.
- It supports team-member collaboration across the process of development, testing, and deployment of multitiered, mixed-workload applications.

WebSphere Studio: Highlights

With WebSphere Studio you can build powerful Java2 Platform, Enterprise Edition (J2EE)technology-compliant applications. These J2EE applications can help reduce the cost and complexity of developing multitier, enterprise services, and can be rapidly deployed and easily enhanced. For example, WebSphere Studio gives your development team access to databases and links to transaction-processing systems. Developers can use the tools they need to create, edit, and validate enterprise application archive files. The development environment in WebSphere Studio provides testing and support for IBM WebSphere Application Server, which allows you to quickly and easily create J2EE applications.

WebSphere Studio reduces development complexity by providing application workflow to help you visually manage the flow of information between application components, Web services, and back-end systems. It also simplifies integration by leveraging a services-oriented architecture that allows developers to interact with all application artifacts in the same way, regardless of their underlying implementations, providing continuity within the development team.

The built-in Web services tools in WebSphere Studio support standards like Simple Object Access Protocol (SOAP), Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL), and Web Services Inspection Language (WSIL).You can construct applications by visually composing components as services.These application services can be accessed through the Web or implemented as local Java services (JavaBeans or Enterprise JavaBeans) or legacy assets. By working with all components as services, you have the flexibility to mix and match functions from different sources, bringing innovative processes, services and value chains to market quickly and efficiently.

In addition to programming, WebSphere Studio enables developers to test and deploy their application from a common environment. Tasks are simplified through the use of the rich set of utilities and wizards, that can significantly improve developer productivity. Studio is based on an open-source project called the Eclipse Workbench - which was originally contributed by IBM. It provides a general platform that enables different tools to share the same look-and-feel and operate in an integrated fashion. The WebSphere Studio Workbench will continually incorporate the features of new releases of the Eclipse Workbench as the open-source community makes them available.

These are some of the highlights of WebSphere Studio:

- It helps reduce the skills needed to develop component-based Web applications.
- It lets developers rapidly create well-structured e-business systems that integrate WebSphere software and traditional transactional environments, including CICS and IMS.
- It promotes the reuse and transformation of existing applications to reduce costs and shorten the development cycle.
- It automates the adoption of industry-standard e-business architecture through visual construction facilities based on open Struts implementation.
- It helps developers create dynamic Web applications, with support for J2EE,XML, and Web services technologies.
- It improves productivity to develop z/OS system-based applications while making the transition to e-business architecture.

- It helps transform and enable developers by facilitating skill transfer and knowledge in mission-critical enterprise technologies.
- It supports team-member collaboration across the process of development, testing and deployment of multitiered, mixed-workload applications.

3.5 WebSphere Studio integration with Informix 4GL

The Informix 4GL language is being enhanced, and 4GL constructs will be incorporated in the Enterprise Generation Language (EGL) to provide sophisticated Internet-based Rapid Application Development (IRAD) capabilities. EGL is also a fourth-generation language, and with it customers can leverage their existing Informix 4GL language skills but with the power of industry-leading WebSphere Studio Integrated Development Environment (IDE) to develop data-driven Web applications. With the integration of EGL and WebSphere Studio, customers will be able to deploy their legacy and modern 4GL applications using both Informix and DB2 databases.

IBM plans to continue supporting Informix 4GL and Four J's customers indefinitely. To accomplish the smooth integration of existing Informix 4GL customers, IBM is enhancing Informix tools and connectivity. To support bringing Informix into the IBM family of products, IBM is working to enable the tools to connect natively with DB2. Along with this closer integration with DB2, IBM plans to incorporate the 4GL language with WebSphere Studio V5. There will also be utilities provided to perform 4GL to EGL conversions to make the task easier for customers.

To support existing customers and development teams, IBM intends to support these initiatives with the latest releases of Informix Dynamic 4GL Four J's Business Development Suite. New releases are expected in 2003 and beyond. IBM will also continue to sell and support Informix SQL (I-SQL) for existing customers for as long as they require. Informix tools and connectivity are being functionally enhanced so they will continue to provide the best support possible to legacy Informix customers.

The Informix 4GL language has been a great benefit to many customers and, by incorporating the language into EGL, IBM hopes to achieve the best of both worlds. IBM is dedicated to maintaining a good relationship with its partner Four J's and plans to continue to promote and support both 4GL-WebSphere and Four J's products as complementary offerings.

4

Installing and configuring WebSphere Studio V5

In Chapter 2, “Installing and configuring IDS V9.40” on page 21 we described how to install and configure IDS. In the current chapter we now provide similar information relative to WebSphere Studio Application Developer (WSAD).

The following topics are discussed:

- **Installing WebSphere Application Developer 5.0 on Windows:** This section provides an overview of the installation process.
- **Configuring WebSphere Studio for use with IDS:** We also verify that WebSphere Studio is configured properly by developing a database Web page that connects to IDS. That Web page enables you to query data from the stores_demo database that is packaged with Informix Dynamic Server. Retrieval of the data verifies that the configuration is correct.
- **Sample application using Database Web Pages:** WSAD makes this sample process easier because it has an Application Server built into it. We will deploy the sample application to the WSAD Application Server.

In Chapter 7, “Working with IDS and WebSphere” on page 143, we will develop a more complex sample application and deploy it to a distributed WebSphere Application Server.

4.1 Installing WSAD on Windows/2000

WebSphere Studio Application Developer can be installed from CD-ROM or an electronic image of WebSphere Studio Application Developer. We installed WSAD from an electronic image that was downloaded from the WebSphere Studio Trial Program site found at:

`http://www7b.boulder.ibm.com/wsdd/downloads/WSsupport.html`

Pre-installation tasks

Before you install the product, check the following items:

- In addition to the space required to install the product, you must have at least 50 MB free on your Windows system drive, and your environment variable TEMP or TMP must point to a valid temporary directory with at least 10 MB free.
- You must not have the IBM HTTP server or WebSphere Application Server, Version 3.5 running.

Install from electronic image: WebSphere Studio Application Developer

There are twelve downloadable parts for WebSphere Studio Application Developer. All twelve parts are self-extracting archives. You must extract the first nine parts and the twelfth part; the other parts are optional.

Steps to install WSAD from electronic image:

1. Log in as an administrator whose ID does not contain double-byte characters.
2. Download all required parts, and any other desired optional parts.
3. Each part is a self-extracting archive. Extract each part into the same temporary directory.
4. From the temporary directory, run setup.exe.
5. The WebSphere Studio Installation Launcher window will open, and it contains several links. Select **Install IBM WebSphere Studio Application Developer**.
6. Follow the on-screen instructions for tasks such as specifying the target installation directory.

Important: Do not install into a directory whose name contains double-byte characters or special characters such as a dollar sign. Doing so may cause undesirable results, such as class path problems, in the WebSphere test environment.

7. Click **Next** and then click **Install** to install WebSphere Studio Application Developer.
8. When WebSphere Studio Application Developer is installed, click **Finish** to close the installation window.

Install from the CD-ROM

You can also install WebSphere Studio Application Developer from the CD-ROM.

Steps to install WSAD from CD-ROM:

1. Insert the CD-ROM into your CD drive.
2. Log in as an administrator whose ID does not contain double-byte characters.
3. Run setup.exe from the root of the CD drive.

Then follow the instructions above for installing from electronic image, starting from step 5.

4.2 Configuring WSAD for IDS

In this section we explain how to configure WebSphere Studio Application Developer for IDS.

Installing JDBC driver on the development server

In Chapter 2, “Installing and configuring IDS V9.40” on page 21, we described how to install IDS 9.40 in a Linux operating environment. The bundle installer also installed the Informix JDBC driver, so if desired, one can also install WebSphere Studio on the Linux server and create and deploy the demos on one single server. This is an interesting approach if you have limited servers available for developing applications, but is not typical.

So, we decided to implement the development environment on a separate Windows server. That being the case, we needed to install the Informix JDBC driver on the Windows server as well, to enable WebSphere Studio on Windows to interact with IDS on the Linux server.

Downloading and installing Informix JDBC driver on the server

Follow these steps to download and install Informix JDBC driver on the server with WebSphere Studio:

1. You can download (get the latest version available) the JDBC driver alone from:

   `http://www-3.ibm.com/software/data/informix/`

2. The installation is very straightforward, but needs a Java Development Kit (you can download it from `http://www.javasoft.com`) installed, and its bin directory needs to be configured in the PATH environment variable. On Windows/2000 you can perform this task by selecting **Start -> Settings -> Control Panel -> System -> Advanced -> Environment Variables**).

   Unzip the downloaded file with WinZip, then double-click the setup.jar file. Then just follow the installation instructions.

Verifying the configuration

Configuration of WebSphere Studio Application Developer for use with IDS can be verified by creating a sample application to access IDS and perform a data retrieval operation. This is how we verified our configuration. It is a very simple application, but is all that is needed at this point. For example, it will be deployed to a WebSphere Application Server for execution. However, it is an Application Server internal to WSAD. The application architecture is depicted in Figure 4-1.

In Chapter 7, “Working with IDS and WebSphere” on page 143, using more complex functions, we will demonstrate how to implement a more rigorous sample application and deploy to a separate WebSphere Application Server.

To access IDS, an application can use a data source / database URL. To test the configuration of WebSphere Studio Application Developer for IDS, we will use this method to develop and test a sample application in WebSphere Studio Application Developer. This application development made easier because we can do it all by using a graphical interface. It will require some knowledge of Java, but does not require a skilled Java programmer.

Architecture of the sample application

The ITSO Stores sample application performs a search operation for a given customer ID, returns the matching customer details, and displays it on the browser screen (Figure 4-1).

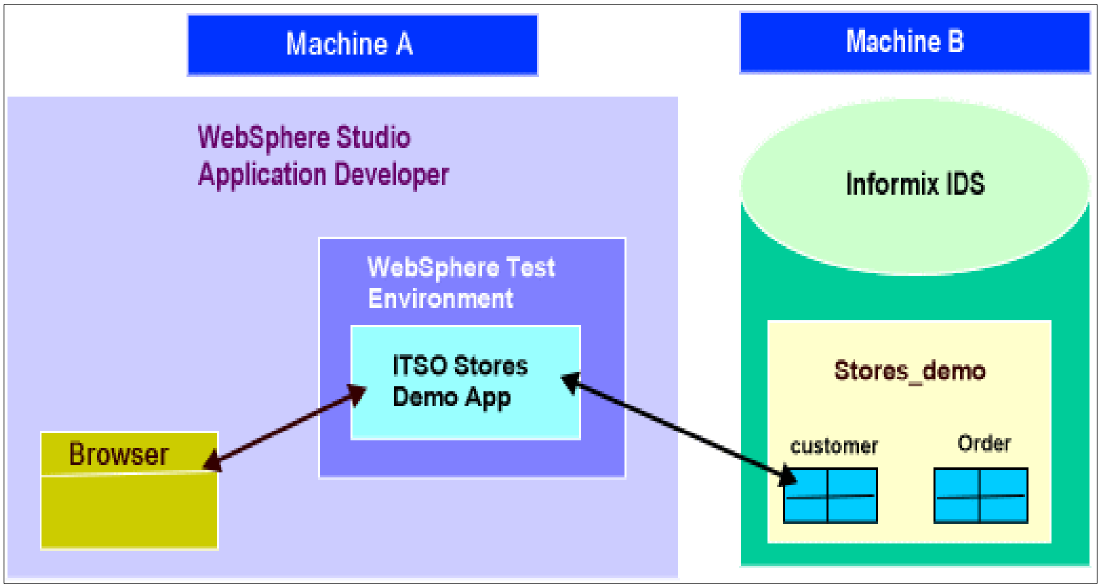


Figure 4-1 Architecture of ITSOStores sample application

4.3 A sample application using Database Web Pages

The section describes the development of a sample application, ITSO Stores, using Database Web Pages, a feature of WSAD. Database Web Pages provide a graphical environment for developing Web applications. These Web applications can be used to perform search, insert, update, and delete operations on database tables. But in this section we will only develop the sample application to perform a ***select*** operation on the data from the stores_demo database table.

However, we will develop another sample application to perform ***select, insert, update, and delete operations***, in Chapter 7, “Working with IDS and WebSphere” on page 143. That sample application will demonstrate the use of additional technologies, such as Java beans, entity beans (EJB), Java Server Pages (JSP), Java Message Services (JMS) and IDS, and Container Managed Persistent Beans. This is to demonstrate more of the robust capabilities of WebSphere and IDS.

Steps to develop a sample application using Database Web Pages

This section steps you through the development of a sample application to perform a *select* operation on the Stores_demo database residing on IDS.

Note: This section takes you through step-by-step instructions to develop a sample application using Database Web Pages. If you prefer not to go through the exercise of developing your own sample application, you can simply download the already developed demo sample application file called *SampleApps.Zip* from Appendix D, “Additional material” on page 343.

Inside the zipped download file are six folders. Open the folder named *SelectApplication*. Inside is the actual sample demo application, called *ITSOStoresDBPages.ear.* First, import this file into your WSAD workspace. Then follow the development instructions in steps 31 to 44 given in the section “A sample application using Database Web Pages” on page 88.

1. Open Web Sphere Studio Application Developer.
2. Open J2EE Perspective (Figure 4-2).

   In the Main Menu, click **Window -> Open Perspective -> Other**.

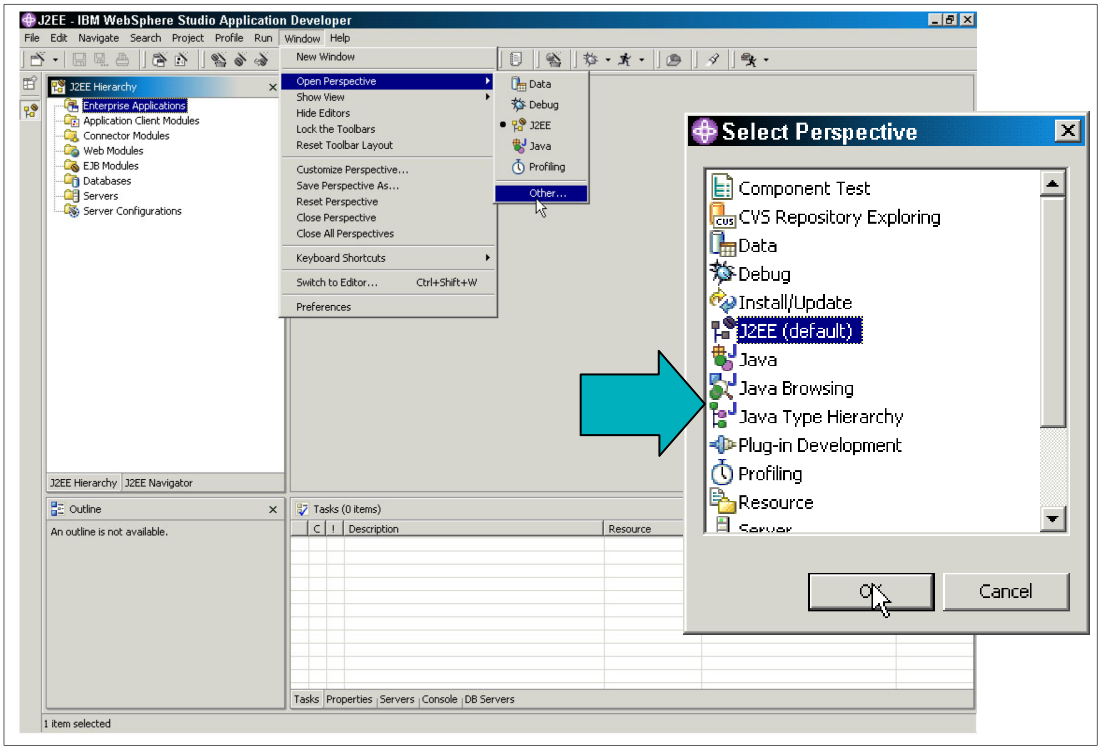
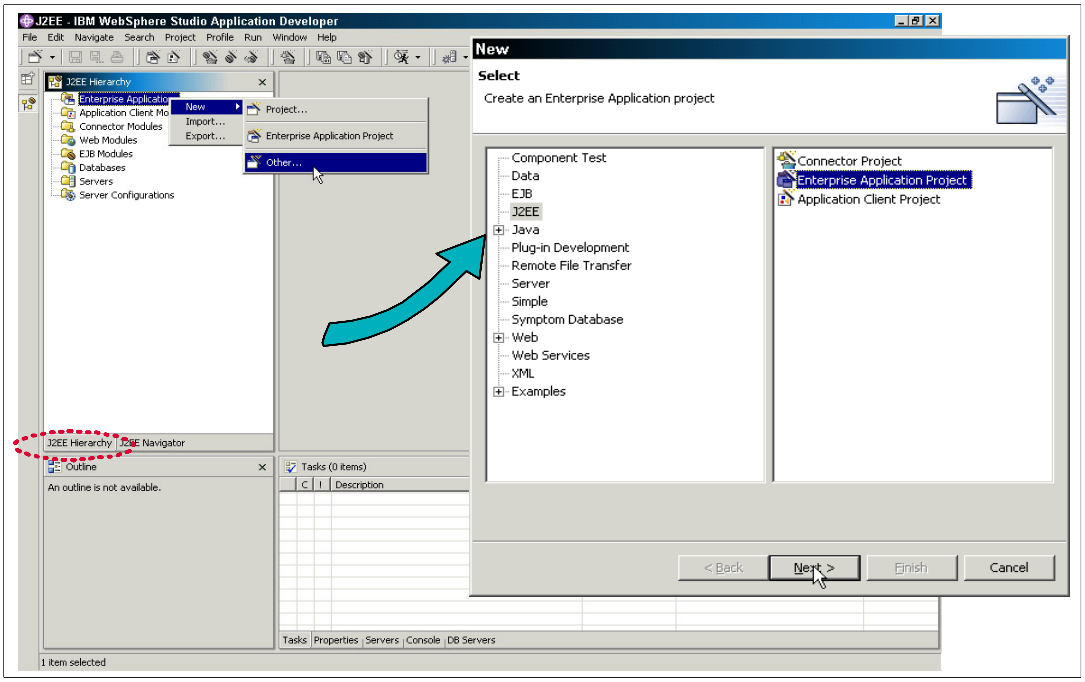


Figure 4-2 Opening a J2EE Perspective

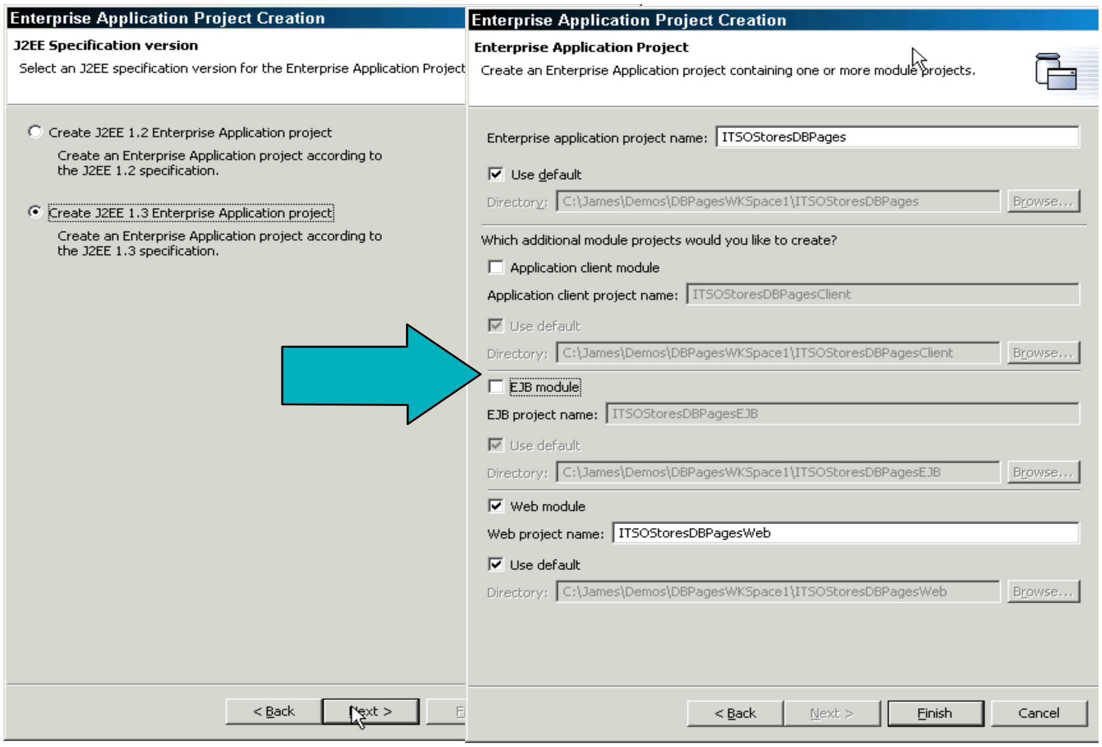
3. A dialogue box pops up to select the perspective, as just shown in Figure 4-2. Select **J2EE(default)** from the list of perspectives. Click the **OK** button and the J2EE Perspective will be opened.
4. Find the **J2EE Hierarchy** view in the left pane of the J2EE Perspective as shown next in Figure 4-3. The J2EE Hierarchy tab is highlighted with a red colored ellipse.
5. Create an Enterprise Application Project and a Web Project:
   a. In J2EE Hierarchy view, right-click the Enterprise Application and then click **New -> Other...** as shown in Figure 4-3. A wizard will pop up to take the parameter value to create a Project.
   b. In the Wizard select **J2EE** from the left pane and you will find various options in the right pane. Select **Enterprise Application Project** from the right pane, then click **Next** as shown in Figure 4-3. A new wizard will be displayed to accept the J2EE Specification Version.
   c. Select **Create J2EE 1.3 Enterprise Application Project** and click the **Next** button. Refer to Figure 4-4.
   d. Enter **Enterprise application project name** as *ITSOStoresDBPages.* Uncheck the **Application client module** and **EJB module**. Make sure that **Web module** is checked. After entering all these details, the wizard should look as shown in Figure 4-4. Click the **Finish** button.
   e. In the **J2EE Hierarchy**, view the two projects that will be created. One is under **Enterprise Applications** and the other is under **Web Modules**.



Select

Create an Enterprise Application project

Figure 4-3 Creating Enterprise Application Project


Enterprise Application Project Creation
< Back
Next >
Enterprise Application Project Creation
Browse...
Browse...
Browse...
Browse...
< Back
Next >
Finish
Cancel

J2EE Specification version

Select an J2EE specification version for the Enterprise Application Project

○ Create J2EE 1.2 Enterprise Application project

Create an Enterprise Application project according to the J2EE 1.2 specification.

● Create J2EE 1.3 Enterprise Application project

Create an Enterprise Application project according to the J2EE 1.3 specification.

Enterprise Application Project

Create an Enterprise Application project containing one or more module projects.

Enterprise application project name: ITSOStoresDBPages

☑ Use default

Directory: C:\James\Demos\DBPagesWKSpace1\ITSOStoresDBPages

Which additional module projects would you like to create?

☐ Application client module

Application client project name: ITSOStoresDBPagesClient

☑ Use default

Directory: C:\James\Demos\DBPagesWKSpace1\ITSOStoresDBPagesClient

☐ EJB module

EJB project name: ITSOStoresDBPagesEJB

☑ Use default

Directory: C:\James\Demos\DBPagesWKSpace1\ITSOStoresDBPagesEJB

☑ Web module

Web project name: ITSOStoresDBPagesWeb

☑ Use default

Directory: C:\James\Demos\DBPagesWKSpace1\ITSOStoresDBPagesWeb

Figure 4-4 Creating Enterprise Application Project

6. Open a Web Perspective.

> **Note:** To open a Web Perspective, follow steps 2 and 3 in the list "Open J2EE Perspective (Figure 4-2)." on page 89. Select **Web** instead of **J2EE(default)** from the list of perspectives. For more information on perspectives, please refer to section 3.1 in the redbook, *Linux Application Development using WebSphere Version 5*, SG24-6431, which can be downloaded from the Redbooks Web site, `http://www.redbooks.ibm.com`.

7. Find Projects ITSOStoresDBPages and ITSOStoresDBPagesWeb in the **J2EE Navigator** view, as shown in Figure 4-5. The J2EE Navigator view is circled with a red ellipse.
8. Expand project **ITSOStoresDBPagesWeb**. Find the WebContent folder under ITSOStoresDBPages.
9. Right-click the **WebConnect** folder and then click **New -> Other**. A wizard will be opened as shown in Figure 4-5.
10. In the wizard, select **Web** from the left pane and then **Database Web Pages** from the right pane. Click the **Next** button as shown in Figure 4-5. The **Database Web Pages** and a wizard will be opened.

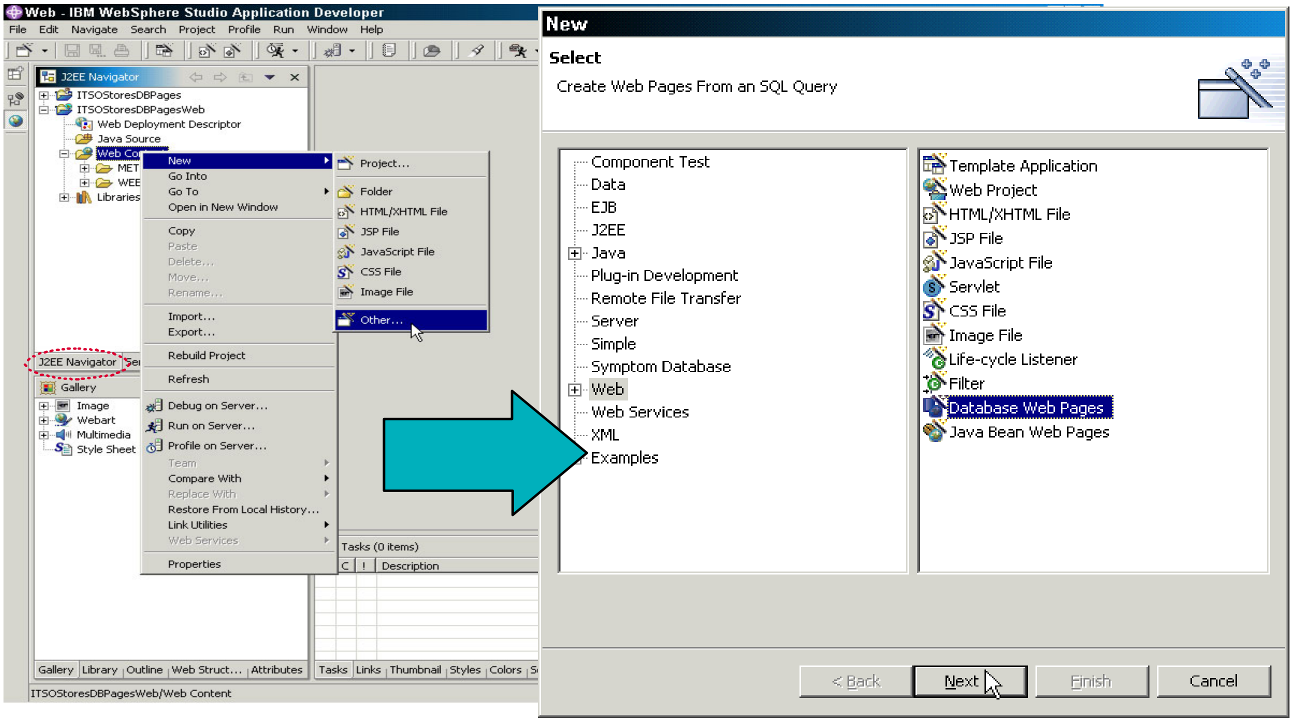
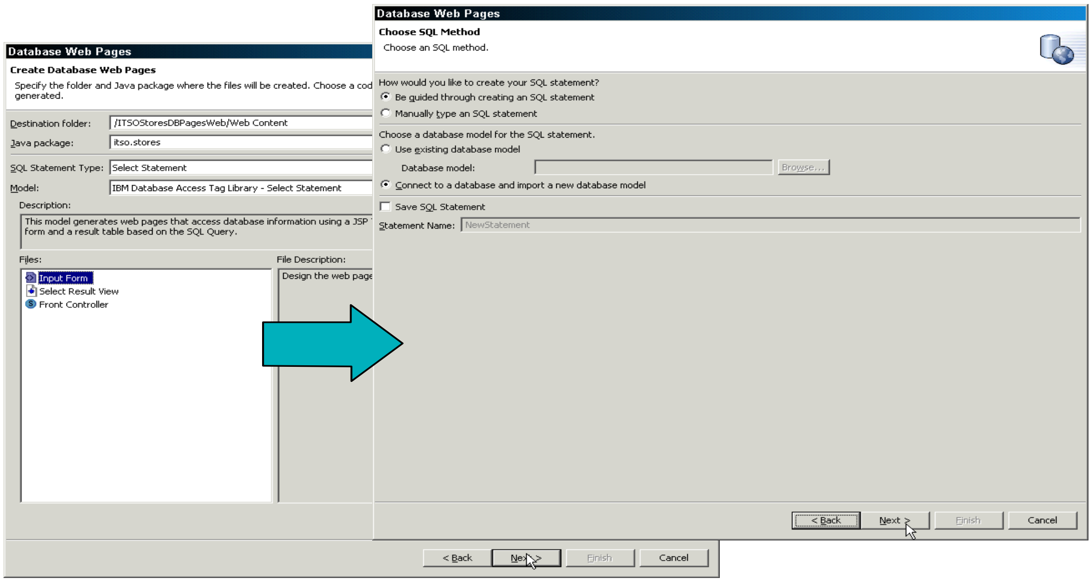


Figure 4-5 Creating Database Web Pages

11. Enter name of the **Java package** as *itso.stores*. Select **SQL Statement Type** as *Select Statement*, **Model** as *IBM Database Access Tag Library - Select Statement*. Cross-check the values from Figure 4-6. Click the **Next** button. The contents of **Data Base Web Pages** Wizard will be changed to **Choose SQL Method**.
12. Select **How would you like to create your SQL statement?** as *Be guided through creating an SQL statement* and **Choose a database model for the SQL statement** as *Connect to a database and import a new database model.* Cross-check the values from the Figure 4-6 Creating Database Web Pages on page 92. Click the **Next** button. The contents of **Data Base Web Pages** Wizard will be changed to **Connection Page**.



Specify the folder and Java package where the files will be created. Choose a cod generated.

Destination folder: /ITSOStoresDBPagesWeb/Web Content

Java package: itso.stores

SQL Statement Type: Select Statement

Model: IBM Database Access Tag Library - Select Statement

Description: This model generates web pages that access database information using a JSP form and a result table based on the SQL Query.

Files: Input Form, Select Result View, Front Controller

File Description: Design the web page

Choose an SQL method.

How would you like to create your SQL statement?

● Be guided through creating an SQL statement

○ Manually type an SQL statement

Choose a database model for the SQL statement.

○ Use existing database model

Database model:

● Connect to a database and import a new database model

□ Save SQL Statement

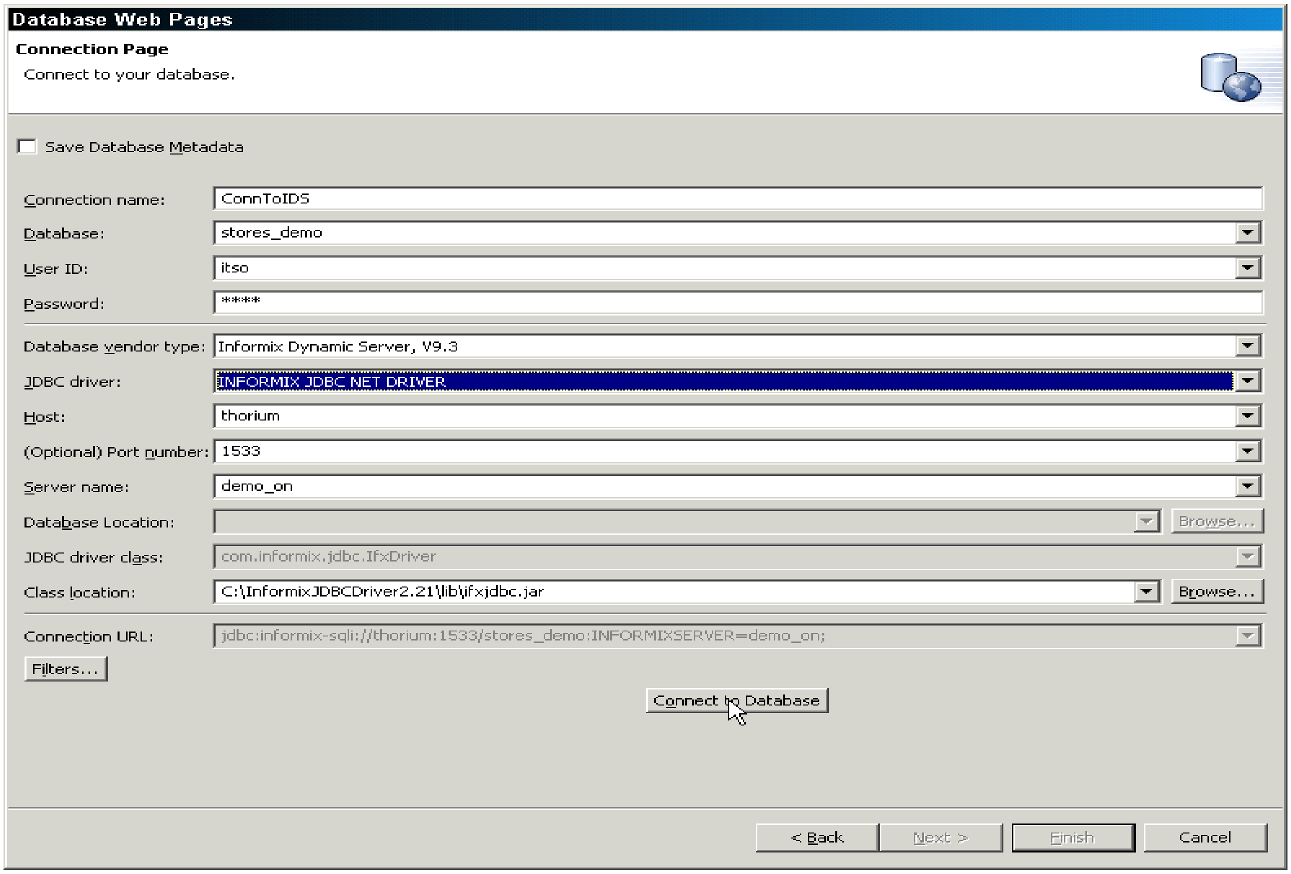
Statement Name: NewStatement

Figure 4-6 Creating Database Web Pages

13. Enter the **Connection name** as *ConnToIDS*, **Database** as *stores_demo,* **User ID** as *itso*, **Password** as password of the itso user. Select **Database vendor type** as *Informix Dynamic Server, V 9.3*, **JDBC driver** as *INFORMIX JDBC NET DRIVER*, **Host** as the ipaddress of the IDS server, **Port number** as port number of the server instance demo_on, **Server name** as *demo_on*, **Class location** as the *<JDBC Driver Installed Directory>\lib\ifxjdbc.jar.* After entering all the values, then cross-check the values with Figure 4-7. Click the button **Connect to Database**. The contents of the **Data Base Web Pages** wizard will be changed to **Construct An SQL Statement**.

Notes:

- To find the port number of the server instance, refer to 13.1.4, “Determining the port number of IDS on Linux” on page 305.
- If you have questions with any of the aforementioned connection parameters, contact your IDS System Administrator.
- If you get an error when you click the button **Connect to Database**, recheck the port number and server name.



□ Save Database Metadata

Connection name: ConnToIDS

Database: stores_demo

User ID: itso

Password: *****

Database vendor type: Informix Dynamic Server, V9.3

JDBC driver: INFORMIX JDBC NET DRIVER

Host: thorium

(Optional) Port number: 1533

Server name: demo_on

Database Location:

JDBC driver class: com.informix.jdbc.IfxDriver

Class location: C:\InformixJDBCDriver2.21\lib\ifxjdbc.jar

Connection URL: jdbc:informix-sqli://thorium:1533/stores_demo:INFORMIXSERVER=demo_on;

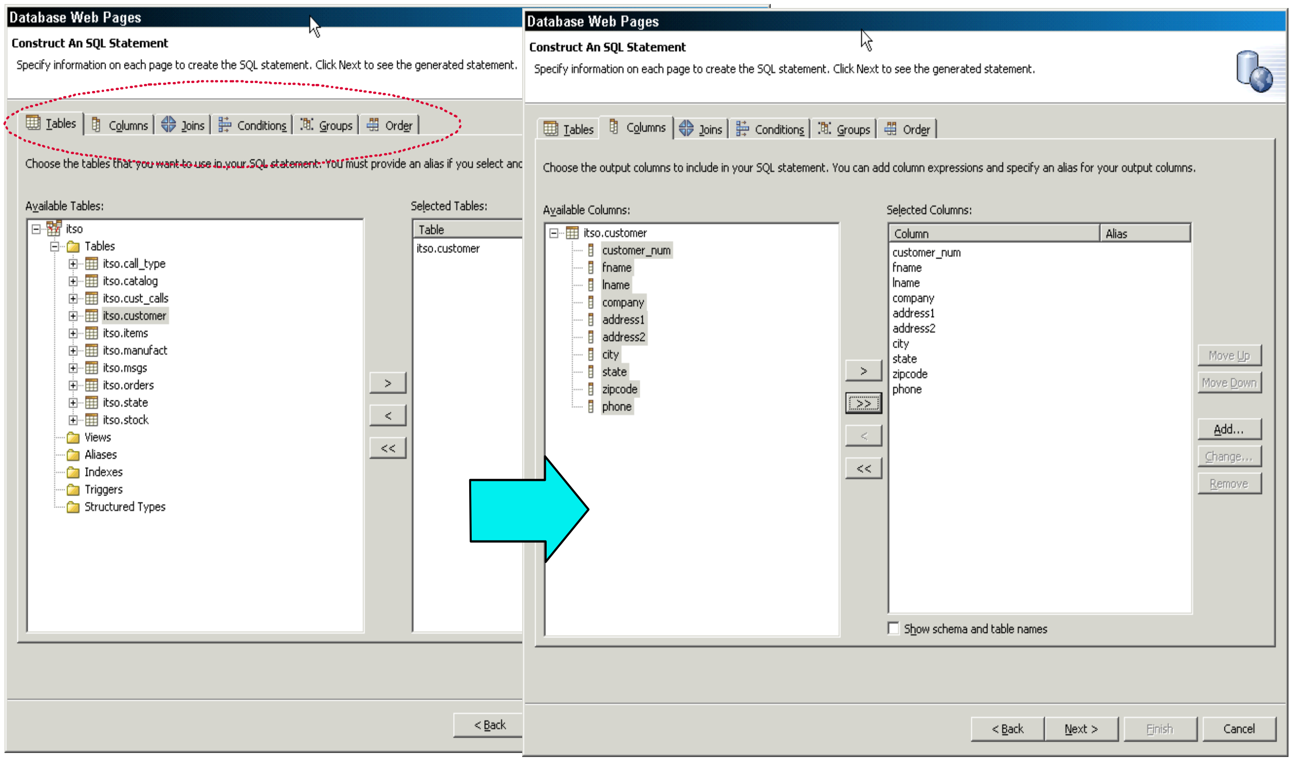
Figure 4-7 Connection to IDS

14. Find the available tables in left pane. Select **itso -> Tables -> itso.customer** and click the button **>**. Find the **itso.customer** table under the Selected Tables list. Refer to Figure 4-8.

15. Click the **Columns** Tab. Here we will select the required columns that we want to see in the output of the search. Select all the columns under the table **itso.customer** and click the **>** button. Find all the columns under the list of **Selected Columns**. Refer to Figure 4-8.

16. Select the tab **Conditions**. Here we can specify the condition to select a record from IDS. We will specify to select a record that matches our customer number.

    Find the columns **Column, Operator, Value, And/Or**. Cells in the first row display combox boxes as soon as you click them. For example, if you click the cell under **Column**, then a combox box with a list of column names will be displayed. From the combox box, select **itso.customer.Customer_num**. Similarly select column **Operator** as **=** , and **Value** as **Build Expression...** For more clarity, please refer to Figure 4-9. To build an expression, the **Expression Builder** wizard will be displayed. We are building this expression to link the customer number of interest to an SQL statement.



Database Web Pages

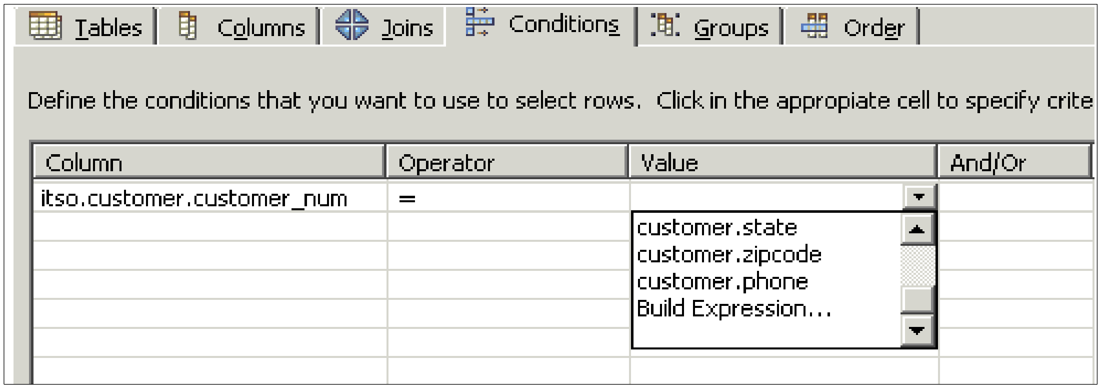
Construct An SQL Statement

Specify information on each page to create the SQL statement. Click Next to see the generated statement.

Choose the tables that you want to use in your SQL statement. You must provide an alias if you select anc

Choose the output columns to include in your SQL statement. You can add column expressions and specify an alias for your output columns.

Figure 4-8 Construct an SQL statement



Define the conditions that you want to use to select rows. Click in the appropiate cell to specify crite

Column	Operator	Value	And/Or
itso.customer.customer_num	=		

Figure 4-9 Conditions tab

17. In the **Expression Builder** wizard, select the type of expression to build as **Constant - numeric, string or host variable,** and click **Next**, as shown in Figure 4-10. The contents of the **Expression Builder** wizard will be changed to **Constant Options Page**.

18. Select Constant type as **String constant** and click **Next**. The contents of the **Expression Builder** wizard will be changed to **String Constant Builder Page** to take the name of the variable which contains customer number entered by the user.

19. Enter the Host variable name as *custNum* and select **String constant type** as the **Host variable name**. Cross-check the values from Figure 4-10. Click the **Finish** button. You will be taken back to the **Database Web Pages** wizard.

20. After finishing the foregoing steps, the contents under the Conditions tab should be as depicted in Figure 4-11. Click the **Next** button in the **Database Web Pages** wizard. The contents of the **Database Web Pages** wizard will be changed to **SQL Statement Page**.

21. In the **SQL Statement Page,** find the SQL statement which we have built. Test the SQL statement by clicking on the button **Execute...** Dialogue box **Execute SQL Statement** will be opened.

22. Click the **Execute** button. Dialogue box **Specify Variable Values** will be opened to accept our customer number. Enter the customer number *102* under the column **Value** by double-clicking in the cell. After entering the customer number press the Tab key and click the **Finish** button as shown in Figure 4-12.

23. You will find the details of customer102, depicted as in Figure 4-12. **Congratulations, you have successfully connected to IDS from WebSphere Studio.** Now we will finish the Web application and deploy it on WebSphere application server.

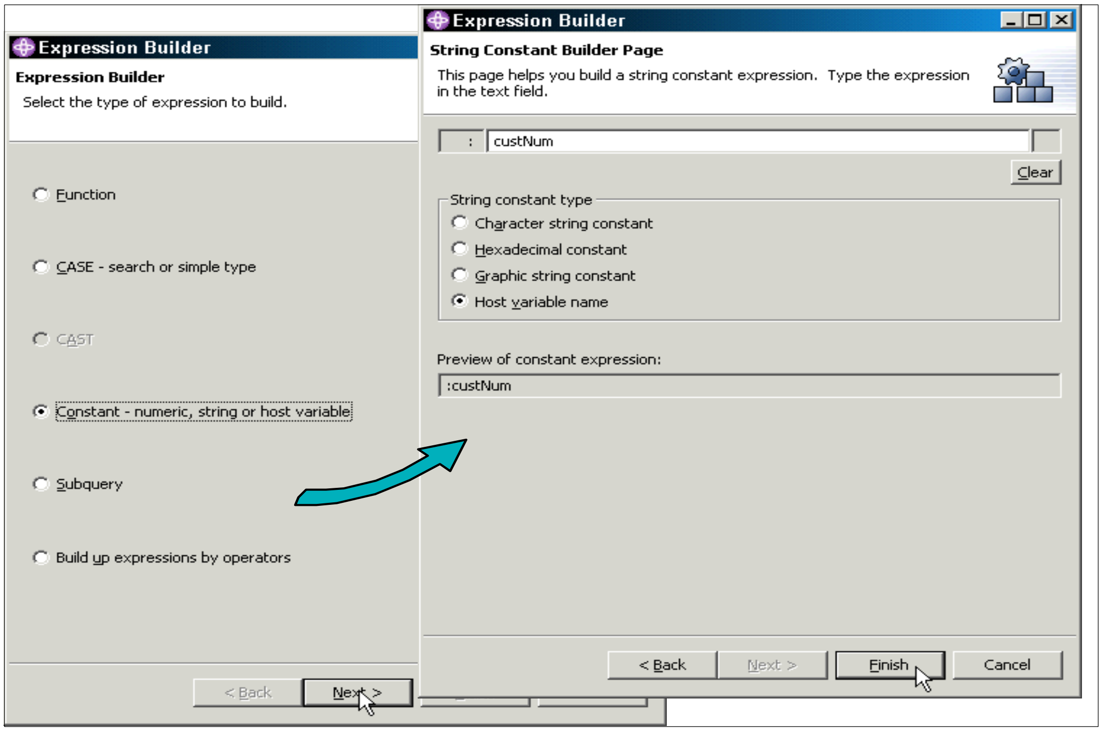
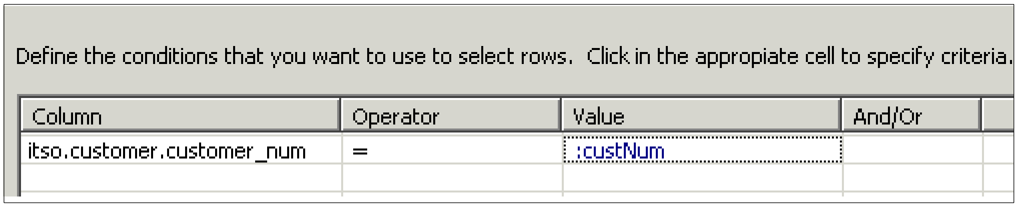

Expression Builder
Expression Builder
Select the type of expression to build.
○ Function
○ CASE - search or simple type
○ CAST
● Constant - numeric, string or host variable
○ Subquery
○ Build up expressions by operators
< Back
Next >
Expression Builder
String Constant Builder Page
This page helps you build a string constant expression. Type the expression in the text field.
:
custNum
Clear
String constant type
○ Character string constant
○ Hexadecimal constant
○ Graphic string constant
● Host variable name
Preview of constant expression:
:custNum
< Back
Next >
Finish
Cancel

Figure 4-10 Host variable


Define the conditions that you want to use to select rows. Click in the appropiate cell to specify criteria.

Column	Operator	Value	And/Or
itso.customer.customer_num	=	:custNum	

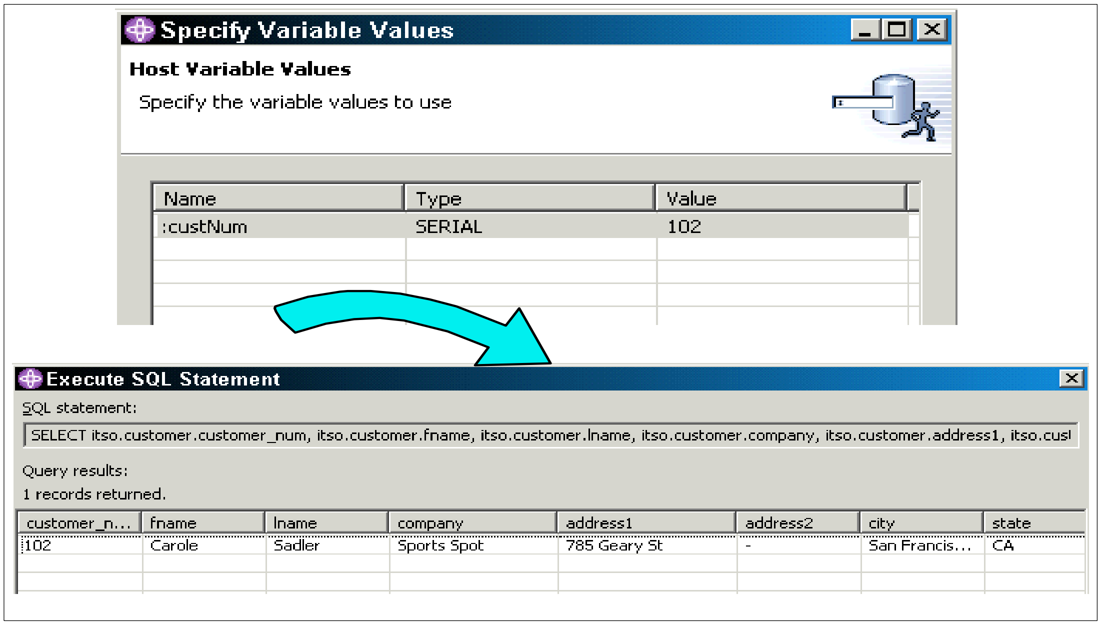
Figure 4-11 Search condition

24. Close the dialogue box **Execute SQL Statement** by clicking the button **Close**.

25. Click the **Next** button in **SQL Statement Page (Database Web Page)**. The contents of **Database Web Pages** wizard will be changed to **Runtime Connection Pages**.

26. Select the database connection as **Use driver manager connection**. Enter **Driver name** as *com.informix.jdbc.IfxDriver,* **URL** as *jdbc:informix-sqli://<hostname>:<portnumber>/stores_demo:INFORMIXSERVER=demo_on;* , **User ID** as *itso*, **Password** as *<password of user itso>,* **Re-enter password** as

<password of user itso>. Click the **Next** button and the contents of the **Database Web Pages** wizard will be changed to **Controller Page**.



Host Variable Values

Specify the variable values to use

Name	Type	Value
:custNum	SERIAL	102

SQL statement:

```
SELECT itso.customer.customer_num, itso.customer.fname, itso.customer.lname, itso.customer.company, itso.customer.address1, itso.cust
```

Query results:

1 records returned.

customer_n...	fname	lname	company	address1	address2	city	state
102	Carole	Sadler	Sports Spot	785 Geary St	-	San Francis...	CA

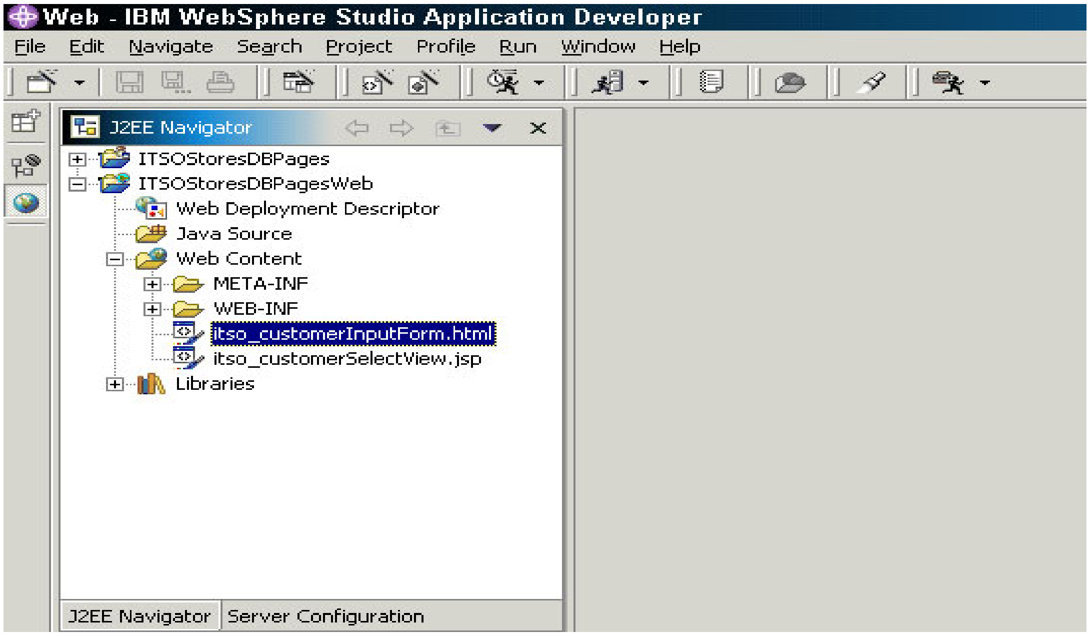
Figure 4-12 Executing SQL statement

27. Select the radio button **Do not use a Front Controller**, then click the **Next** button. The contents of the **Database Web Pages** wizard will be changed to **Design the Input Form.** This form is used to specify our interested customer number.

28. Click the **Next** button and the contents of the **Database Web Pages** wizard will be changed to **Design the Select View**. This form is used to specify the interested columns to display in the result page. Select all Result set columns from the left pane.

29. Click the **Next** button and the contents of the **Database Web Pages** wizard will be changed to **Specify Prefix**. Click the **Finish** button.

30. WebSphere will generate the itso_customerInputForm.html and itso_customerSelectView.jsp in the Web Content folder in the project **ITSOStoresDBPagesWeb**. Refer to Figure 4-13. Development of our application is finished. Now we need to configure the test environment for our application.


Web - IBM WebSphere Studio Application Developer
File Edit Navigate Search Project Profile Run Window Help
J2EE Navigator
J2EE Navigator
Server Configuration

```
ITSOStoresDBPages
ITSOStoresDBPagesWeb
    Web Deployment Descriptor
    Java Source
    Web Content
        META-INF
        WEB-INF
        itso_customerInputForm.html
        itso_customerSelectView.jsp
    Libraries
```

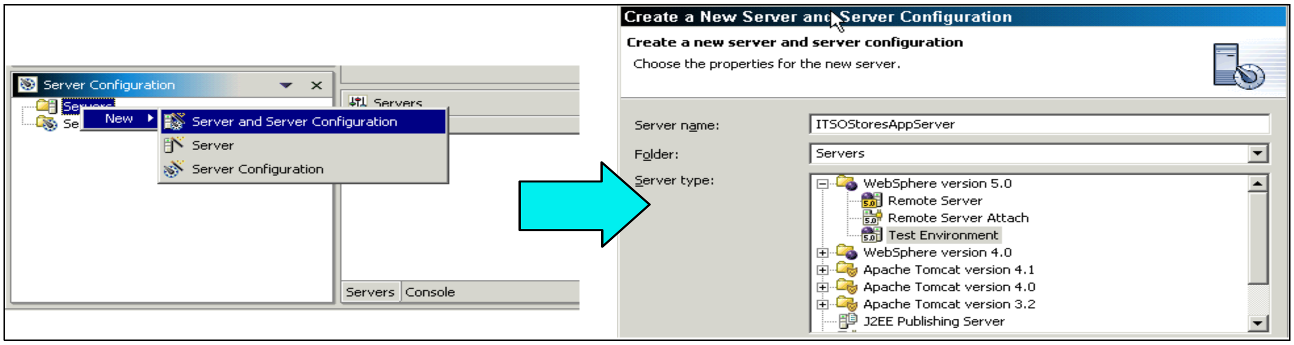
Figure 4-13 Files created by Database Web Pages

Creating a new Server Instance and Server Configuration:

31. Select the Server perspective (**Window -> Open Perspective -> Server**).

32. In Server Configuration view, select **Servers** and right-click the it. Select **New** -> **Server and Server Configuration**. The **Create a New Server and Server Configuration** wizard will pop up. Refer to Figure 4-14.

33. Enter the **Server name** as *ITSOStoresAppServer*, **Folder** as *Servers*, **Server type** as *Test Environment.* Keep things as default and click the **Next** button. The contents of **Create a New Server and Server Configuration** wizard will be changed to **WebSphere Server Configuration settings**. Refer to Figure 4-14.



Create a New Server and Server Configuration

Create a new server and server configuration

Choose the properties for the new server.

Server name: ITSOStoresAppServer

Folder: Servers

Server type:

```
WebSphere version 5.0
    Remote Server
    Remote Server Attach
    Test Environment
WebSphere version 4.0
Apache Tomcat version 4.1
Apache Tomcat version 4.0
Apache Tomcat version 3.2
J2EE Publishing Server
```

Figure 4-14 Creating a new Server

34. Enter **HTTP port number** as *9080*, and click the **Finish** button. Wait for some time and the Server and Server Configuration will be created.

35.Find the created server **ITSOStoresAppServer**, under **Servers** in Server Configuration view. The View should be changed to **Advanced**, to see the existing server configurations.

36.Click the drop-down menu of the Server Configuration view as shown in Figure 4-15. Click **View -> Advanced**. Find the **ITSOStoresAppServer** under **Server Configurations** as highlighted with the red ellipse in Figure 4-15. **Congratulations, you have successfully finished the creation of Server Instance and Server Configuration.**

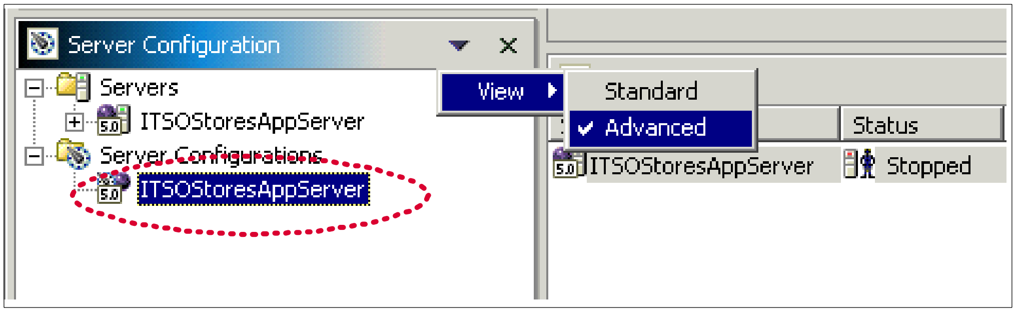
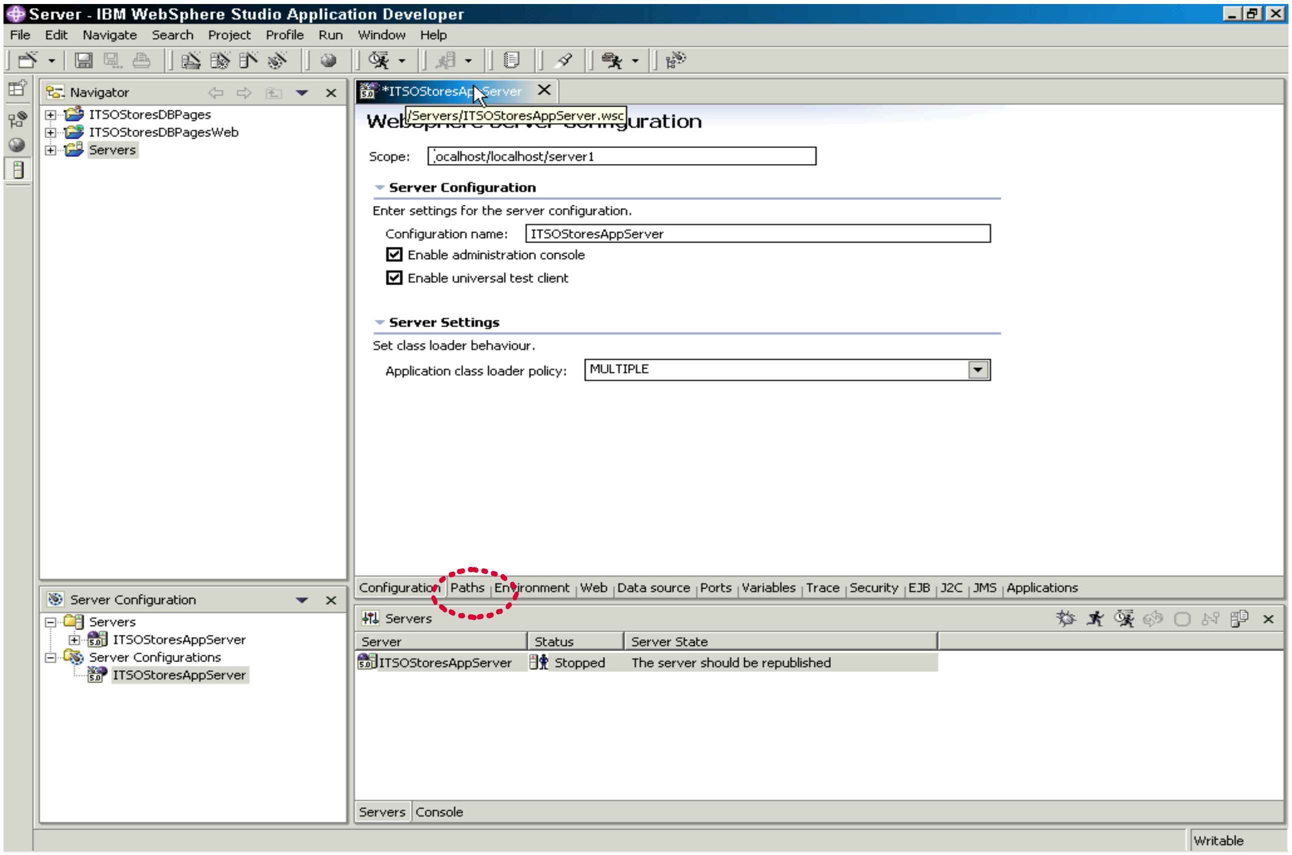


Figure 4-15 Server Configuration view

Configuring WebSphere Test Environment (WebSphere Studio) for IDS:

37.Open **ITSOStoresAppServer** server configuration, by double-clicking on it. It will be opened as shown in Figure 4-16. Find the various tabs called **Configuration**, **Paths**, **Environment**, **Web**, and **Data source**.

38.Go to the **Paths** tab that is highlighted in Figure 4-16. Add files <JDBC Driver Installed Dir>\lib\ifxjdbc.jar and <JDBC Driver Installed Dir>\lib\ifxjdbcx.jar to classpath, by clicking on the button **Add External Jars**, under Class Path as shown in Figure 4-17. Save and close the **ITSOStoresAppServer** configuration. **Congratulations, you have successfully finished the Configuration of WebSphere Studio for IDS**. Now we need to deploy the application to our test server, **ITSOStoresAppServer**.


Server - IBM WebSphere Studio Application Developer
File Edit Navigate Search Project Profile Run Window Help
Navigator
ITSOStoresDBPages
ITSOStoresDBPagesWeb
Servers
*ITSOStoresAppServer
/Servers/ITSOStoresAppServer.wsc
WebSphere Server Configuration
Server Configuration
Servers
ITSOStoresAppServer
Server Configurations
ITSOStoresAppServer
Configuration | Paths | Environment | Web | Data source | Ports | Variables | Trace | Security | EJB | J2C | JMS | Applications
Servers | Console
Writable

Scope: ocalhost/localhost/server1

Server Configuration

Enter settings for the server configuration.

Configuration name: ITSOStoresAppServer

☑ Enable administration console

☑ Enable universal test client

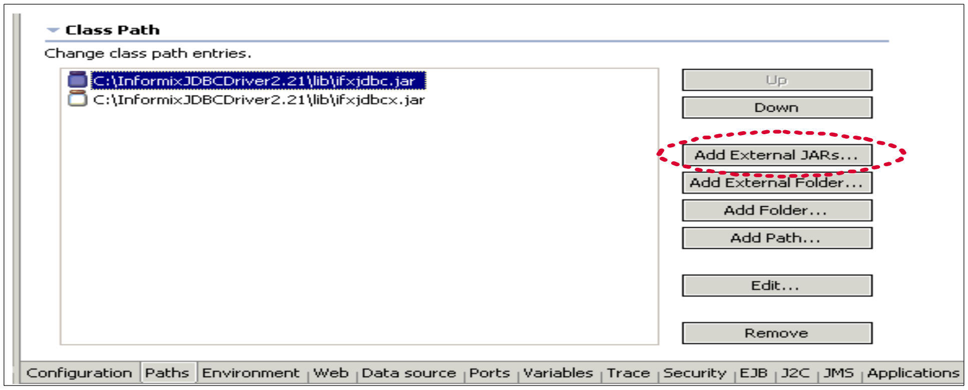
Server Settings

Set class loader behaviour.

Application class loader policy: MULTIPLE

Server	Status	Server State
ITSOStoresAppServer	Stopped	The server should be republished

Figure 4-16 Server Configuration


Up
Down
Add External JARs...
Add External Folder...
Add Folder...
Add Path...
Edit...
Remove
Configuration | Paths | Environment | Web | Data source | Ports | Variables | Trace | Security | EJB | J2C | JMS | Applications

Class Path

Change class path entries.

C:\InformixJDBCDriver2.21\lib\ifxjdbc.jar

C:\InformixJDBCDriver2.21\lib\ifxjdbcx.jar

Figure 4-17 Server Classpath

Deploying the Web application in the test server:

39. In the Server Configuration View, select **ITSOStoresAppServer** under Server Configurations and right-click the it. We will now add our application ITSOStoresDBPages EAR file. Select **Add -> ITSOStoresDBPages**, as shown in Figure 4-18. **ITSOStoresDBPages** will be deployed in test server **ITSOStoresAppServer**.

Starting the test server:

40.In the **Servers** view, find the server **ITSOStoresAppServer**. Right-click it and select **Start** as shown in Figure 4-19. The server will take a few minutes to start. In the **Servers** view, status of server, **ITSOStoresAppServer** will be changed to started.

Running the ITSOStoresDBPages application on an application server:

41.Go to Web Perspective (**Window -> Open Perspective -> Web**)

42.In the **J2EE Navigator** view, right-click the file ITSOStoresDBPagesWeb -> Web Content -> itso_customerInputForm.html, and select **Run on Server,** as shown in Figure 4-20.

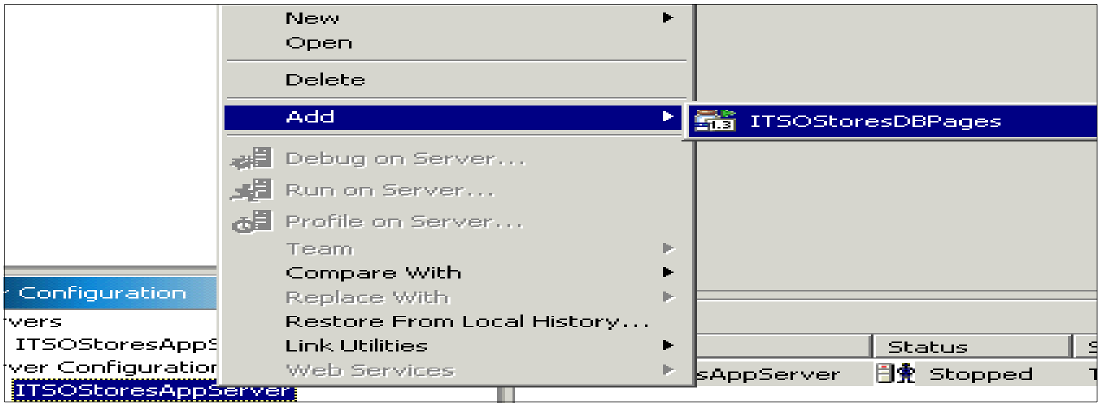


Figure 4-18 Deploying the application

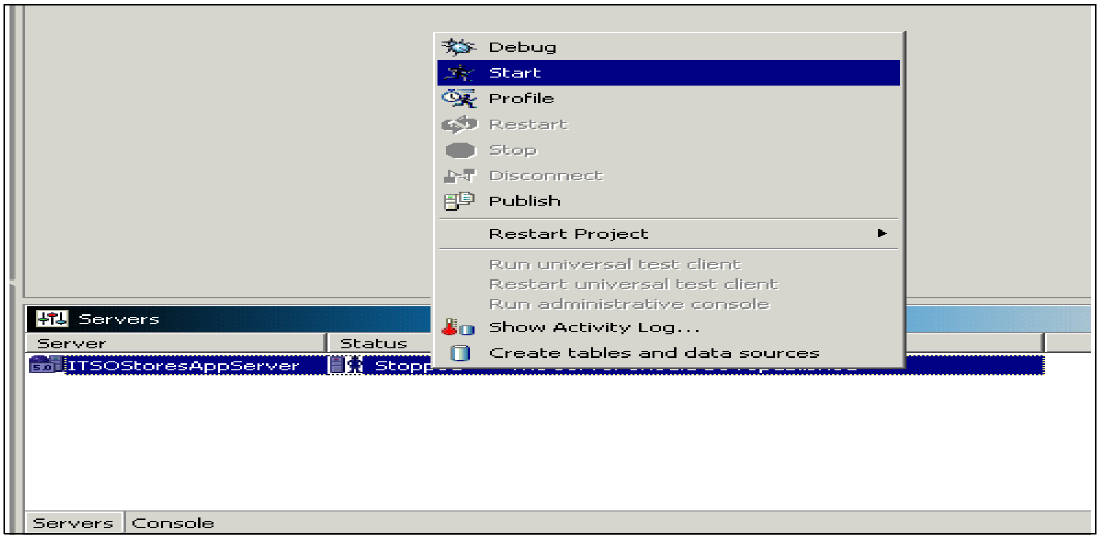


Figure 4-19 Starting the test server

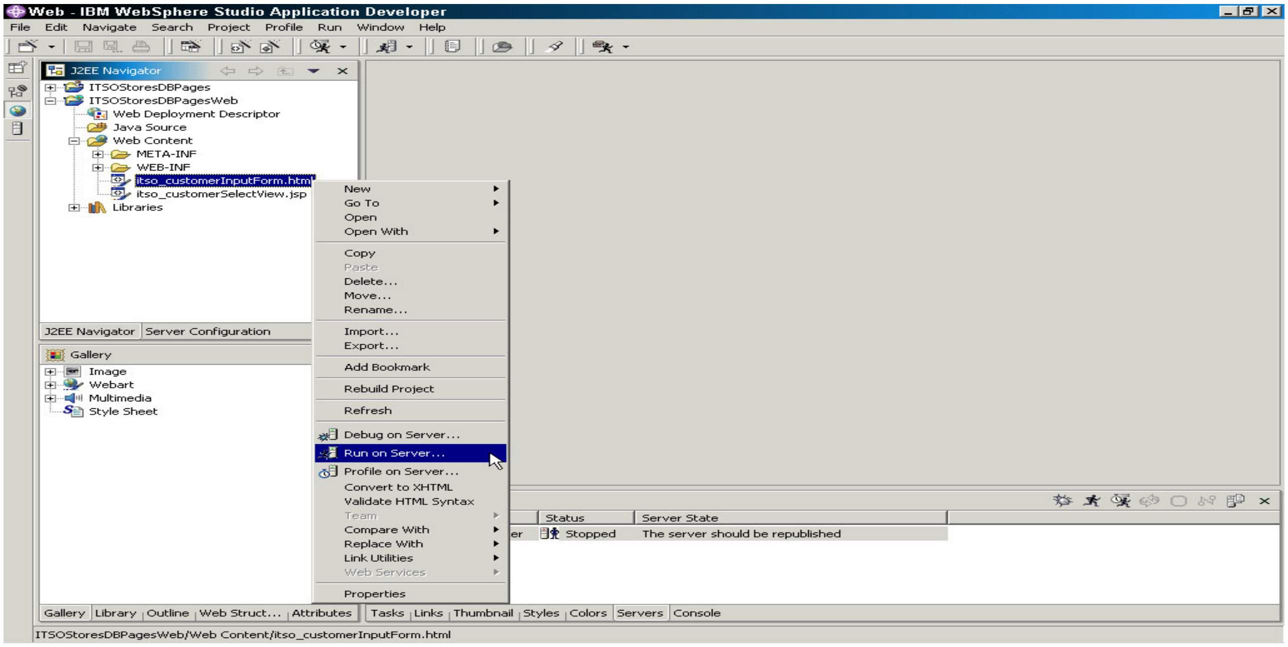
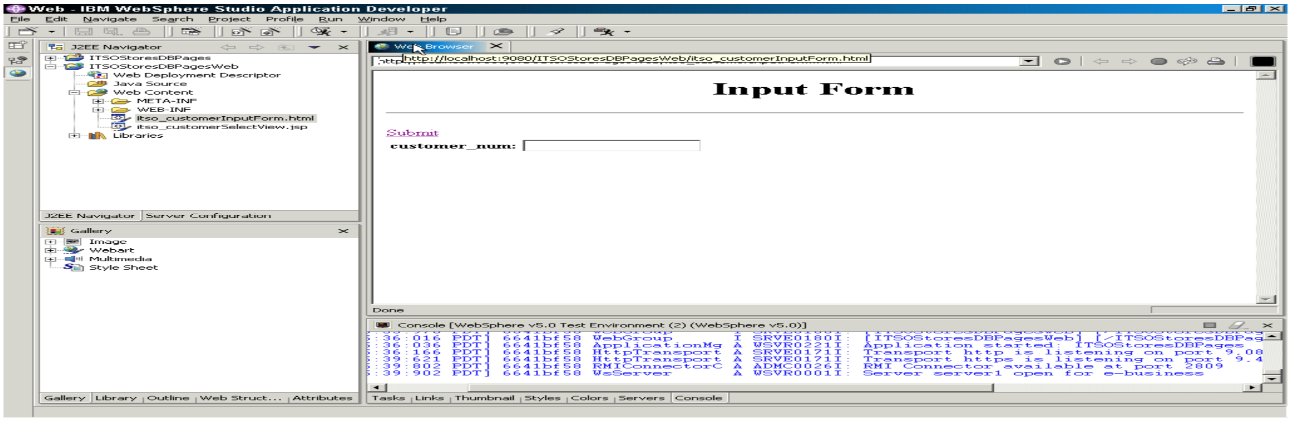

Web - IBM WebSphere Studio Application Developer
File Edit Navigate Search Project Profile Run Window Help
J2EE Navigator
ITSOStoresDBPages
ITSOStoresDBPagesWeb
Web Deployment Descriptor
Java Source
Web Content
META-INF
WEB-INF
itso_customerInputForm.html
itso_customerSelectView.jsp
Libraries
New
Go To
Open
Open With
Copy
Paste
Delete...
Move...
Rename...
Import...
Export...
Add Bookmark
Rebuild Project
Refresh
Debug on Server...
Run on Server...
Profile on Server...
Convert to XHTML
Validate HTML Syntax
Team
Compare With
Replace With
Link Utilities
Web Services
Properties
J2EE Navigator | Server Configuration
Gallery
Image
Webart
Multimedia
Style Sheet
Status
Server State
Stopped
The server should be republished
Gallery | Library | Outline | Web Struct... | Attributes
Tasks | Links | Thumbnail | Styles | Colors | Servers | Console
ITSOStoresDBPagesWeb/Web Content/itso_customerInputForm.html

Figure 4-20 Invoking the application

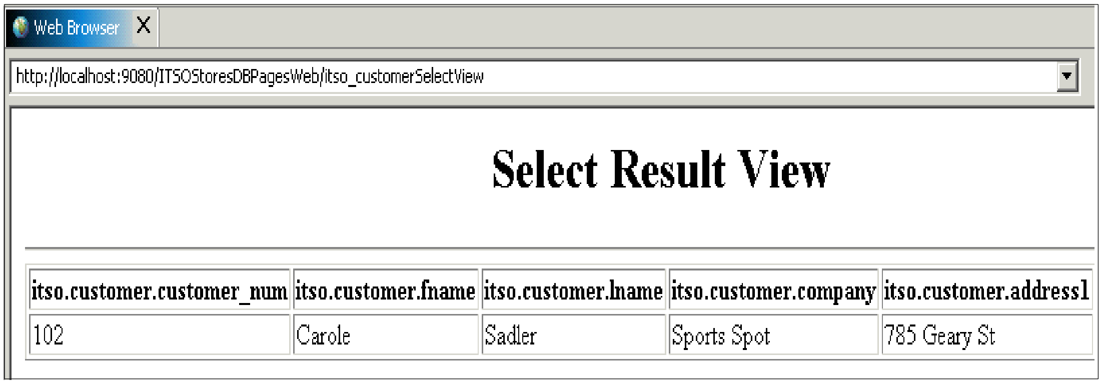
43. A page entitled **Input Form** will be displayed, as shown in Figure 4-21. Now our application is ready to do search operations on IDS and give us details on the customers of interest.


Web - IBM WebSphere Studio Application Developer
File Edit Navigate Search Project Profile Run Window Help
J2EE Navigator
ITSOStoresDBPages
ITSOStoresDBPagesWeb
Web Deployment Descriptor
Java Source
Web Content
META-INF
WEB-INF
itso_customerInputForm.html
itso_customerSelectView.jsp
Libraries
J2EE Navigator | Server Configuration
Gallery
Image
Webart
Multimedia
Style Sheet
Web Browser
http://localhost:9080/ITSOStoresDBPagesWeb/itso_customerInputForm.html
Input Form
Submit
customer_num:
Done
Console [WebSphere v5.0 Test Environment (2) (WebSphere v5.0)]
Gallery | Library | Outline | Web Struct... | Attributes
Tasks | Links | Thumbnail | Styles | Colors | Servers | Console

```
:36:016 PDT] 6641bf58 WebGroup         I SRVE0180I: [ITSOStoresDBPagesWeb] [/ITSOStoresDBPag
:36:036 PDT] 6641bf58 ApplicationMg    A WSVR0221I: Application started: ITSOStoresDBPages
:36:166 PDT] 6641bf58 HttpTransport    A SRVE0171I: Transport http is listening on port 9,08
:39:621 PDT] 6641bf58 HttpTransport    A SRVE0171I: Transport https is listening on port 9,4
:39:802 PDT] 6641bf58 RMIConnectorC    A ADMC0026I: RMI Connector available at port 2809
:39:902 PDT] 6641bf58 WsServer         A WSVR0001I: Server server1 open for e-business
```

Figure 4-21 Input Form

44. Enter **customer_num** as *102*, and click the **Submit** link. The **Select Result View** page will be displayed with the customer details of customer number 102, as shown in Figure 4-22. **Congratulations, you have successfully finished the testing of the ITSOStoresDBPages application in WebSphere Studio**. Our next step is to deploy the application in the WebSphere Application Server. This will be discussed in “Deploy the ITSOStoresDBPages application in WAS” on page 115.


Web Browser
http://localhost:9080/ITSOStoresDBPagesWeb/itso_customerSelectView

Select Result View

itso.customer.customer_num	itso.customer.fname	itso.customer.lname	itso.customer.company	itso.customer.address1
102	Carole	Sadler	Sports Spot	785 Geary St

Figure 4-22 Customer Results

4.3.1 Deploying the application in WebSphere Application Server

As previously mentioned, we will deploy this sample application to an application server that is internal to WSAD. This keeps the exercise very simple and easy. In Chapter 7, “Working with IDS and WebSphere” on page 143, we will develop a more complex application and deploy it to WebSphere Application Server.

To deploy an application in any WebSphere Application Server (even this WSAD internal application server), it should be bundled as a **EAR** file. From WSAD, export the application as an EAR file.

Here are the steps to deploy the application in WAS:

45. In WSAD, go to **J2EE** perspective (**Window -> Open Perspective -> J2EE**).
46. Right-click **Enterprise Applications -> ITSOStoresDBPages.** Click the **Export** button and a wizard will be displayed.
47. In Export Wizard, select **EAR file** and click **Next**. The contents of the **Export** wizard will be changed to **EAR Export**.
48. Select **What resources do you want to export?** as *ITSOStoresDBPages* and **Where do you want to export resources to?** as *C:\Demos\ITSOStoresDBPages.ear* (Any temporary directory). Click the **Finish** button. The ITSOStoresDBPages.ear file will be exported to *C:\Demos\*.

4.3.2 Summary

In this chapter, we developed the sample application and tested it with WebSphere Studio. We also deployed the application to a ***TEST*** application server, within WebSphere Studio. In the next chapter, we will describe how to install and configure a ***REAL*** WebSphere Application Server. Then, in subsequent chapters, we will develop a more rigorous application and deploy it on the real WebSphere Application Server.

5

Installing and configuring WebSphere Application Server

In previous chapters we have described the installation and configuration of IDS on Linux and WSAD on Windows. Now we need to complete our environment by installing the WebSphere Application Server. We have used the Application Server for exercises in Chapter 4, “Installing and configuring WebSphere Studio V5” on page 85, for example, but it was an application server internal to WebSphere Studio Application Developer (WSAD).

In the current chapter we provide step-by-step instructions for installing and configuring the WebSphere Application Server (WAS) product.

We cover the following topics:

- Installing WebSphere Application Server 5.0, Enterprise Edition on Linux.
- Configuring WebSphere Application Server for IDS.
- Administration Repository in WebSphere Application Server 5.0.

5.1 Installing WAS on SuSE Linux V8.0

Prior to WebSphere Application Server 5.0 Version, one of the prerequisites to install WebSphere Application Server was to have a database, which was used as administration repository for WebSphere Application Server. In WebSphere Application Server Version 5.0, that has changed. Now XML configuration files are used as the administration repository, thus removing the dependency of WAS on a database. However, a database is still needed, for example, to store the session information.

Pre-installation tasks

Before installing the IBM WebSphere Application Server V5.0 Enterprise Edition for Linux, a few pre-installation checks and tasks need to be completed:

- Create embedded JMS server user and groups
- Check IP ports are unused
- Stop Web server processes

Create the embedded JMS server user and groups:

The installation program does not automatically create the embedded JMS server user and groups, but it does check to see if the prerequisite user and groups for embedded messaging are configured. If they have not been created, the installation of WebSphere Application Server will stop and display an error message indicating that this needs to be done.

> **Note:** WebSphere will not install if the required user and groups for the embedded messaging are not configured, and you choose the embedded JMS installation option.

You need to create the mqm user and two groups, mqm and mqbrkrs. The user you define to control WebSphere (root in our case) and the mqm user must be members of both groups. To do this requires some familiarity with Linux commands. We used Linux SuSE V8 for our implementation.

Follow these steps to create the embedded JMS server user and groups:

1. Log in as *root* and start a terminal session.

2. Create two new groups:

```
groupadd mqm
groupadd mqbrkrs
```

3. Create a user mqm and add it to the mqm and mqbrkrs groups.

```
useradd -g mqm -G mqbrkrs -m mqm
```

4. Add the *root* user to the mqm and mqbrkrs groups.

- Open the file /etc/group with the command:
  – vi /etc/group
- search for mqm. Find the entries as follows:
  – mqm:x:500:mqm
  – mqbrkrs:x:501:mqm
- Add root user to groups mqm and mqbrkrs, by editing the file /etc/group as follows:
  – mqm:x:500:mqm,root
  – mqbrkrs:x:501:mqm,root

5. Log off and then on again to get the new permissions.

Check that IP ports are unused:

Check to see that there are no existing active services that use the following IP ports on the server:

- 80 (HTTP)
- 443 (HTTPS - optional)
- 2809 (Bootstrap port)
- 9080 (HTTP transport)
- 9443 (HTTPS transport - optional)
- 9090 (HTTP Administrative Console port)
- 5557 (Internal JMS server)
- 5558 (Internal JMS server)
- 5559 (Internal JMS server)
- 7873 (Data replication service port)
- 8880 (SOAP connector port)

We suggest using the following command for this task:

```
netstat -l
```

Installing WebSphere Application Server V5.0

This section describes the installation of WebSphere Application Server. It has step-by-step instructions to help you accomplish the task with minimum configuration effort.

> **Note:** Instead of using the interactive installation procedure described here, you could optionally elect to install WebSphere using the automated or silent installation procedure. See the WebSphere InfoCenter article “Installing silently” for details at the following URL:
>
> http://publib7b.boulder.ibm.com/wasinfo1/en/info/aes/ae/tins_runSilent.html

To install WebSphere Application Server on Linux, follow the steps outlined below:

Typically WebSphere Application Server is distributed on CD-ROMs, but trial versions are also distributed as ***.tar*** files. There are no major differences in the installation of WebSphere Application Server from CD-ROM or .tar files. But for clarity, both alternatives are presented below.

WAS Installation from tar file:

1. Log in as *root* and start a terminal session
2. Go to the directory where the ***.tar*** file is located. Decompress the file using the following command.

```
tar -xvf ./WAS5.0_Enterprise.tar
```

After decompressing the file, you will find a directory called linuxi386 has been created.

3. Go the directory linuxi386 and find the file named LaunchPad.sh

After finishing the steps described above, please read the ***Attention*** note below and then continue at Step 4.

WAS Installation from CD-ROM:

1. Log in as root and start a terminal session.
2. Insert and mount the WebSphere Application Server V5.0 CD-ROM, using the command:

   mount /mnt/cdrom

> **Attention:** If you are running X-windows with either the Gnome or KDE window manager, the CD-ROM may automatically be mounted for you. To verify this, use the command:
>
> ```
> mount | grep /mnt/cdrom
> ```
>
> If you receive any output, that means the CD-ROM has already been mounted.

3. Go to CD-ROM directory.

   ```
   cd /mnt/cdrom
   ```

> **Attention:** To install WebSphere Application Server on SuSE Version 8.0, we need to comment out two lines in LuanchPad.sh. Open LaunchPad.sh using the following command
>
> **vi LaunchPad.sh** and comment as shown below
>
> #!/bin/sh
>
> **#LD_ASSUME_KERNEL=2.2.5**
>
> **#export LD_ASSUME_KERNEL**
>
> LPDIR=`dirname $0`
>
> cd $LPDIR
>
> ./jdk/java/bin/java -jar launchpad.jar
>
> For more details, please refer to Chapter 13, “Implementation hints and tips” on page 295.

4. After reading the ***Attention*** note above, start the WebSphere Application Server LaunchPad.

   ```
   ./LaunchPad.sh
   ```

5. In the language selection window, select **English** and click **OK**.
6. In the WebSphere Application Server LaunchPad window, click **Install the product**, as shown in Figure 5-1.

> **Tip:** On slower servers, it might take a few seconds for the application to respond when you click **Install the product**. Please be patient, as multiple clicks will launch multiple installation programs.

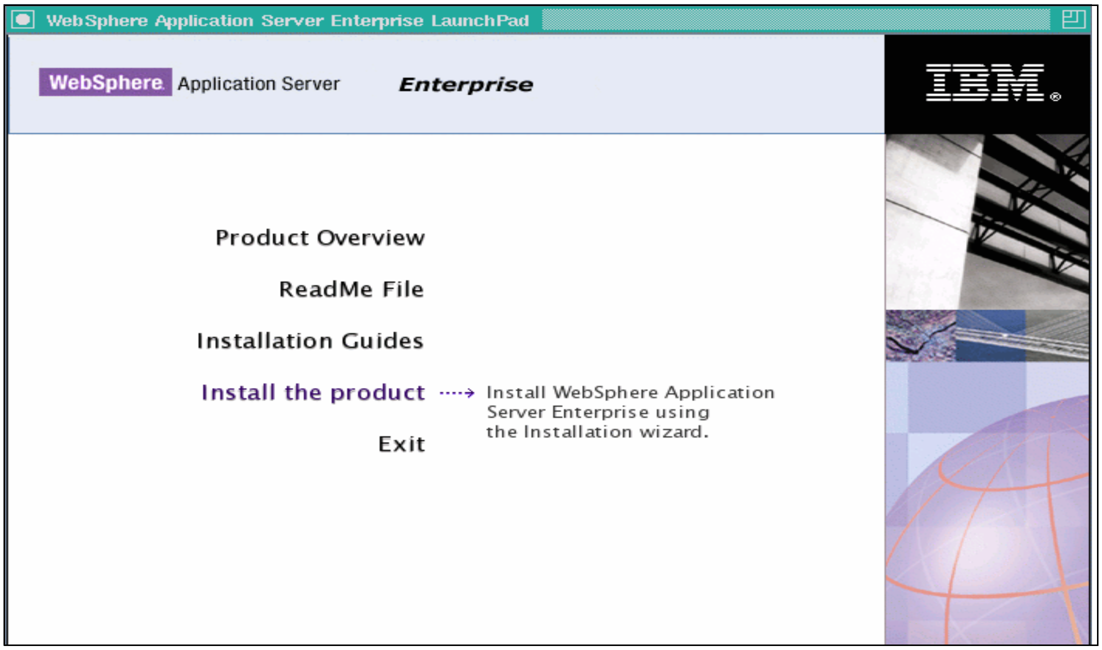
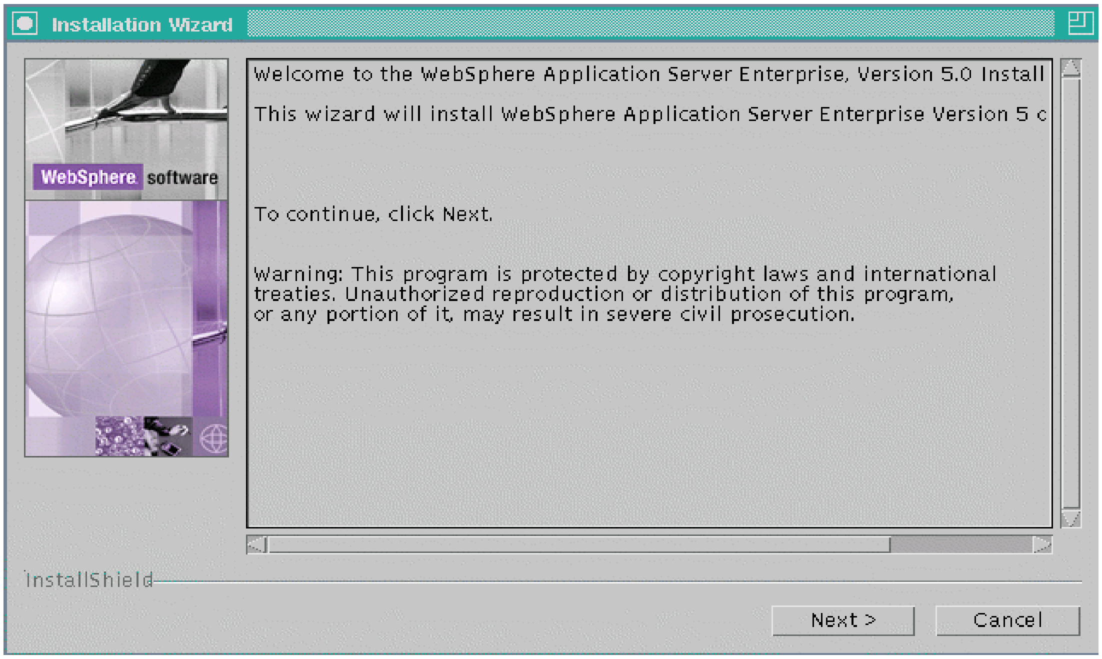

WebSphere Application Server Enterprise LaunchPad
WebSphere Application Server Enterprise
IBM
Product Overview
ReadMe File
Installation Guides
Install the product
Install WebSphere Application Server Enterprise using the Installation wizard.
Exit

Figure 5-1 WebSphere Application Server, Launch Pad

7. In the Wizard language selection window, select **English** and click **OK**.
8. In the Welcome to WebSphere Application Server window click **Next** to continue, as shown in Figure 5-2.



Welcome to the WebSphere Application Server Enterprise, Version 5.0 Install

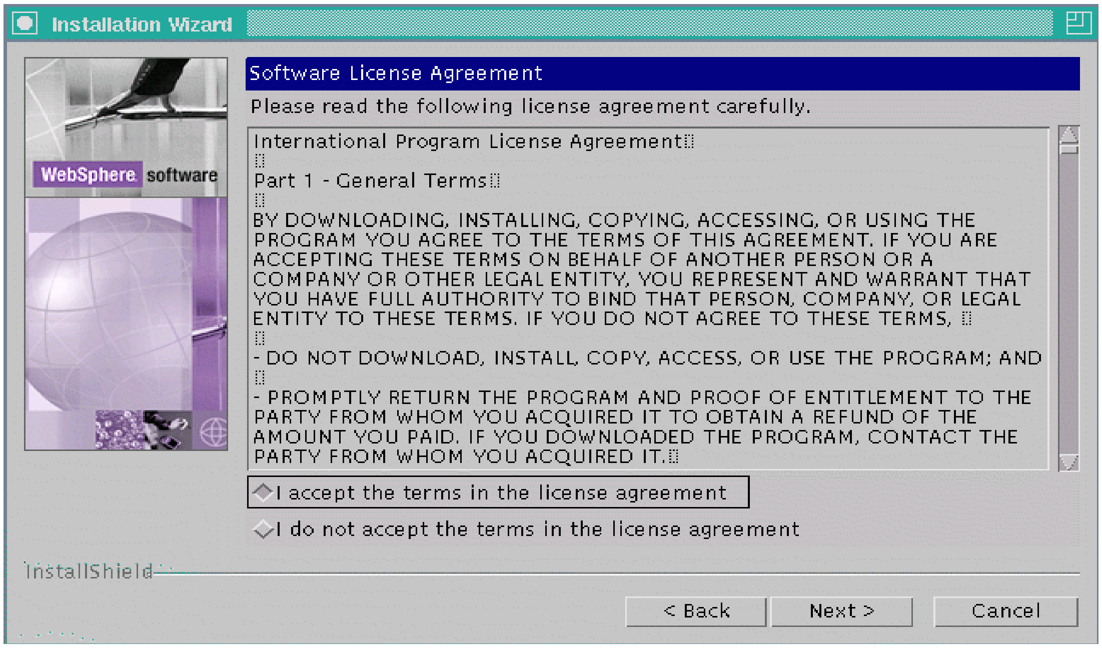
This wizard will install WebSphere Application Server Enterprise Version 5 c

To continue, click Next.

Warning: This program is protected by copyright laws and international treaties. Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil prosecution.

Figure 5-2 Welcome to WAS

9. In the software license agreement window, select **I accept the terms in the license agreement** and click **Next** to agree to the terms of the agreement, as shown in Figure 5-3.



Software License Agreement

Please read the following license agreement carefully.

International Program License Agreement

Part 1 - General Terms

BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE PROGRAM YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,

- DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE PROGRAM; AND

- PROMPTLY RETURN THE PROGRAM AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM YOU ACQUIRED IT TO OBTAIN A REFUND OF THE AMOUNT YOU PAID. IF YOU DOWNLOADED THE PROGRAM, CONTACT THE PARTY FROM WHOM YOU ACQUIRED IT.

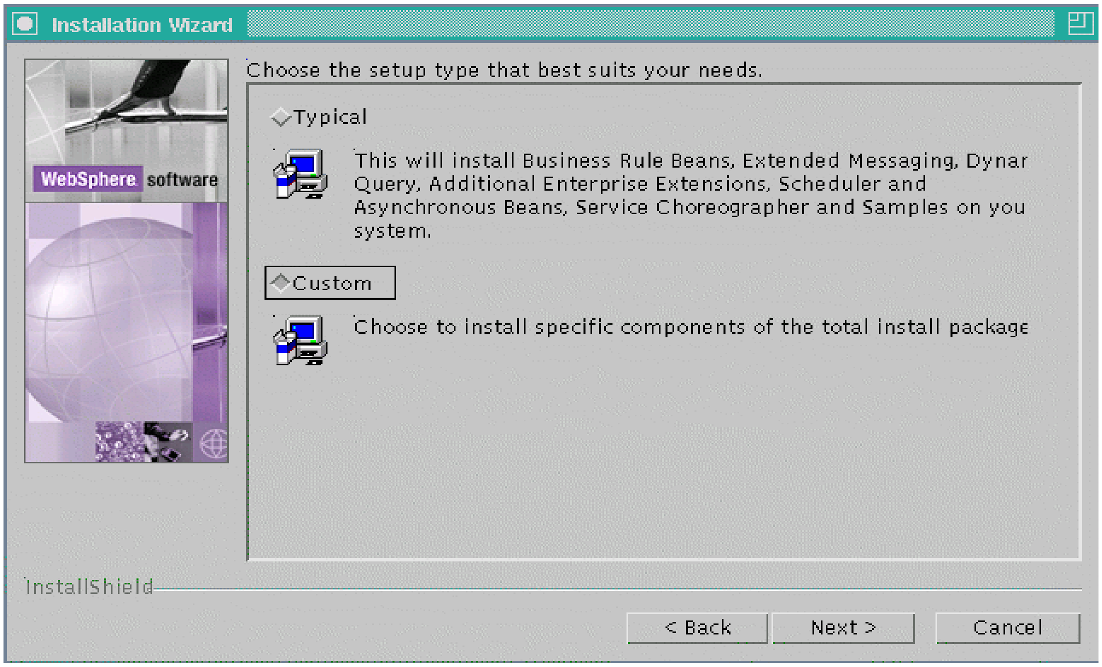
◆ I accept the terms in the license agreement

◇ I do not accept the terms in the license agreement

Figure 5-3 Software License Agreement

The installation Wizard will check the system prerequisites, then display the setup type window:

10. In the setup type window, select the **Custom** setup type option, as shown in Figure 5-4, and click **Next**.



Choose the setup type that best suits your needs.

◇ Typical

This will install Business Rule Beans, Extended Messaging, Dynar Query, Additional Enterprise Extensions, Scheduler and Asynchronous Beans, Service Choreographer and Samples on you system.

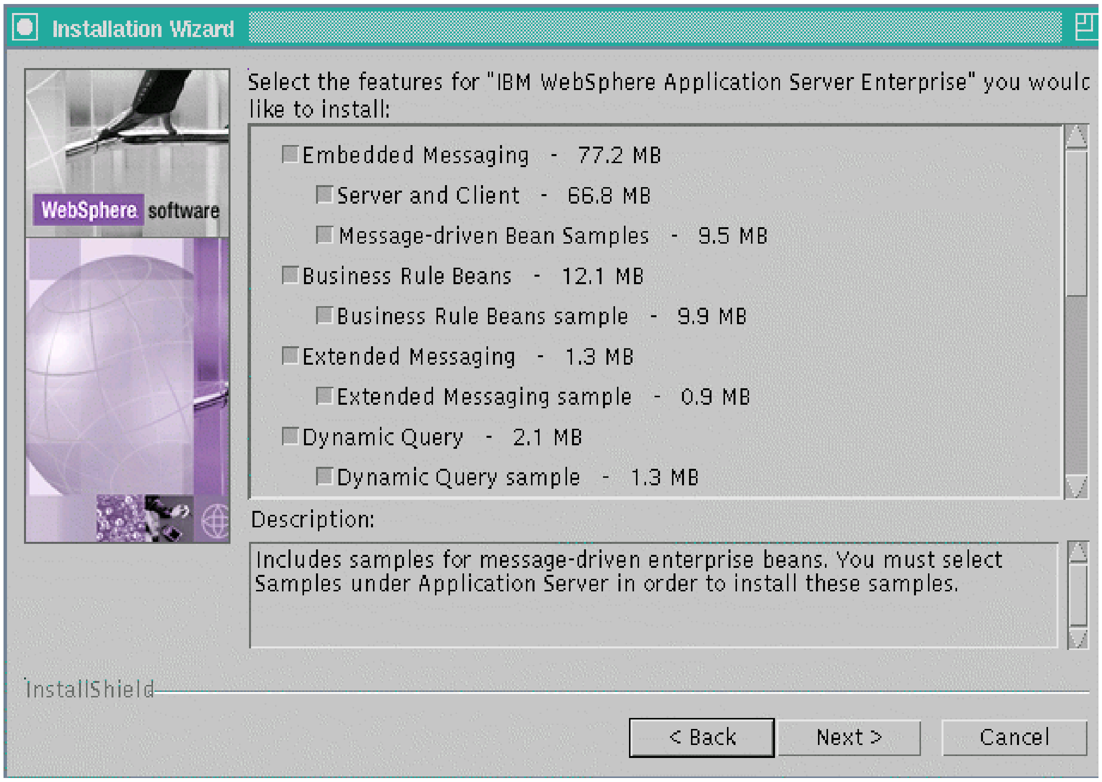
◆ Custom

Choose to install specific components of the total install package

Figure 5-4 Setup types

11. In the **setting the installation directories** window, you can either accept the default, or change the installation path for the WebSphere Application Server. Click **Next**.

12. In the next window, take a look at the different features of WebSphere Application Server as shown in Figure 5-5. By default, all the required features will be selected to be installed. Then click **Next**.



Select the features for "IBM WebSphere Application Server Enterprise" you woulc like to install:

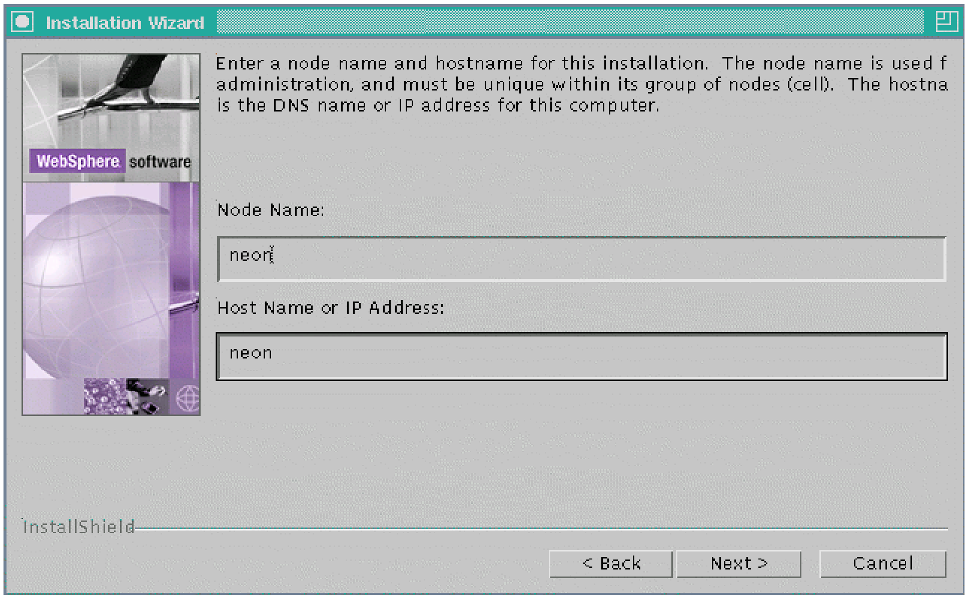
- □ Embedded Messaging - 77.2 MB
  - □ Server and Client - 66.8 MB
  - □ Message-driven Bean Samples - 9.5 MB
- □ Business Rule Beans - 12.1 MB
  - □ Business Rule Beans sample - 9.9 MB
- □ Extended Messaging - 1.3 MB
  - □ Extended Messaging sample - 0.9 MB
- □ Dynamic Query - 2.1 MB
  - □ Dynamic Query sample - 1.3 MB

Description:

Includes samples for message-driven enterprise beans. You must select Samples under Application Server in order to install these samples.

Figure 5-5 Features of WAS 5.0

13. In the **node name** and **host name** window, enter the node name and host name for your installation. In our case we set the node name and host name to **neon**, as shown in Figure 5-6. Click **Next**.



Enter a node name and hostname for this installation. The node name is used f
administration, and must be unique within its group of nodes (cell). The hostna
is the DNS name or IP address for this computer.

Node Name:

neon

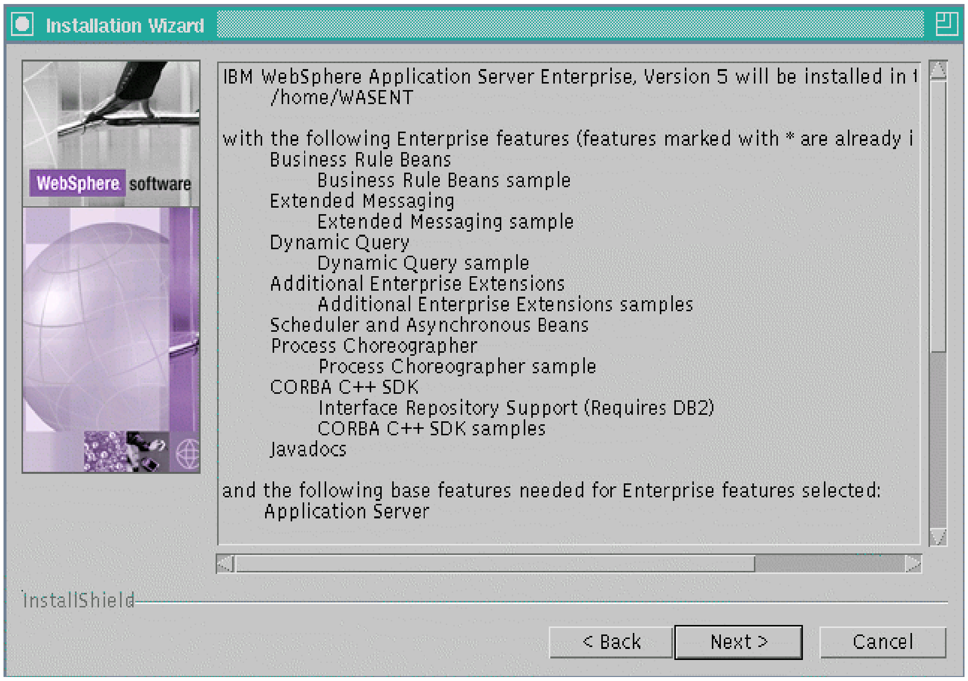
Host Name or IP Address:

neon

Figure 5-6 Node Name and Host Name

14. In the next window, verify your installation settings as shown in Figure 5-7, and click **Next** to start the installation.

The installation of IBM WebSphere Application Server V5.0 will commence. Depending on the speed of your server, the install will finish in only a few minutes.

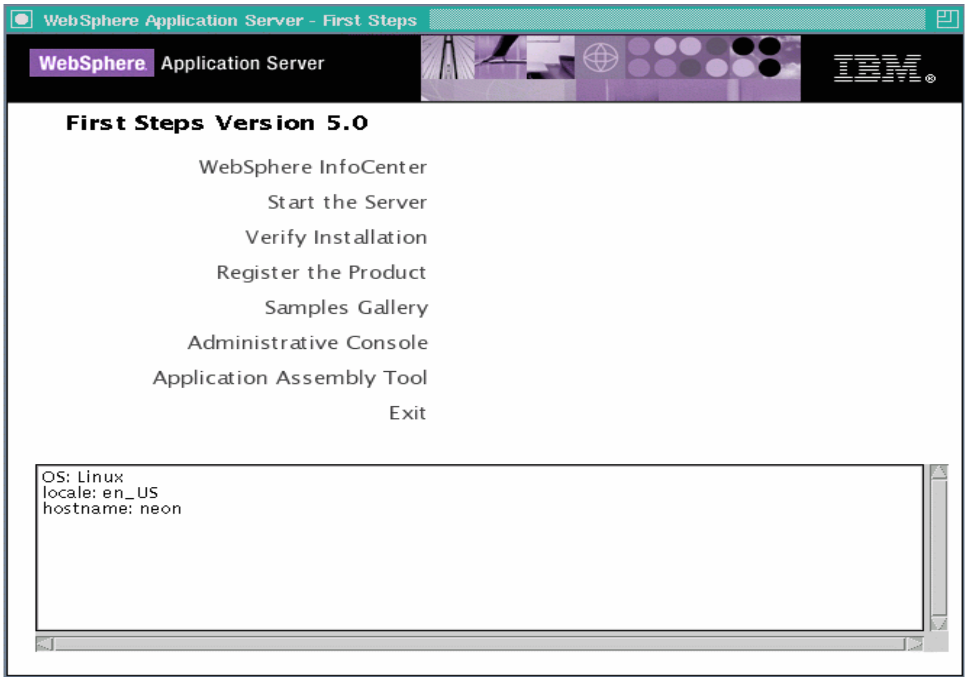


```
IBM WebSphere Application Server Enterprise, Version 5 will be installed in t
        /home/WASENT

with the following Enterprise features (features marked with * are already i
        Business Rule Beans
                Business Rule Beans sample
        Extended Messaging
                Extended Messaging sample
        Dynamic Query
                Dynamic Query sample
        Additional Enterprise Extensions
                Additional Enterprise Extensions samples
        Scheduler and Asynchronous Beans
        Process Choreographer
                Process Choreographer sample
        CORBA C++ SDK
                Interface Repository Support (Requires DB2)
                CORBA C++ SDK samples
        Javadocs

and the following base features needed for Enterprise features selected:
       Application Server
```

Figure 5-7 Summary of Installation settings

15. Once the installation is complete, click **Finish** in the Installation Wizard finished window.

After the successful installation of IBM WebSphere Application Server V5.0, the window shown in Figure 5-8 will automatically start. If for some reason the First Steps window does not appear, it can be started from the command line by running the script <WAS_HOME>/bin/firststeps.sh.



```
OS: Linux
locale: en_US
hostname: neon
```

Figure 5-8 First Steps

5.1.1 Verifying the installation

To verify that WebSphere Application Server V5.0 was installed correctly, click **Verify Installation** in the First Steps window. This command will attempt to start the WebSphere Application Server. During the startup process, additional information will be displayed in the log pane at the bottom of the First Steps window. Alternatively, the Verify Installation utility can also be started from the command line as follows:

```
cd <WAS_HOME>/bin/ivt.sh
```

The output should look similar to the list in Example 5-1.

Example 5-1 Output from ivt script

```
IVTL0095I: defaulting to host appsrv11 and port 9080
IVTL0010I: Connecting to the WebSphere Application Server appsrv11 on port: 9080
IVTL0020I: Could not connect to Application Server, waiting for server to start
IVTL0025I: Attempting to start the Application Server osName = Linux
IVTL0030I: Running /opt/WebSphere/AppServer/bin/startServer.sh server1
>ADMU0116I: Tool information is being logged in file
><WAS_HOME>logs/server1/startServer.log
>ADMU3100I: Reading configuration for server: server1
>ADMU3200I: Server launched. Waiting for initialization status.
>ADMU3000I: Server server1 open for e-business; process id is 1400
IVTL0050I: Servlet Engine Verification Status - Passed
IVTL0055I: JSP Verification Status - Passed
IVTL0060I: EJB Verification Status - Passed
IVTL0070I: IVT Verification Succeeded
IVTL0080I: Installation Verification is complete
```

Tip: To start and stop the application server, two shell scripts are available in the <WAS_HOME>/bin directory.

`startServer.sh`

The command to start server1 is:

```
<WAS_HOME>/bin/startServer.sh server1
```

`stopServer.sh`

The command to stop server1 is:

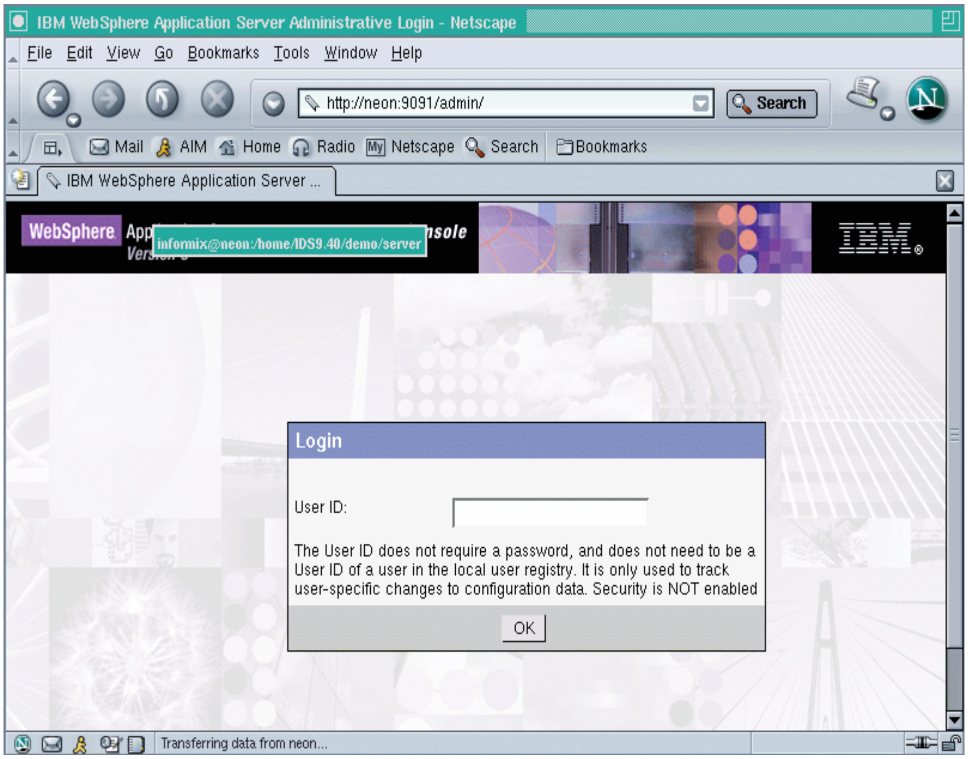
```
<WAS_HOME>/bin/stopServer.sh server1
```

Application server up and running

Once WebSphere Application Server is up and running, you can launch the Administrative Console. You have a choice to either launch the Administrative Console from the First Steps window, shown in Figure 5-8, or by entering the URL in your browser:

- To Launch Administrative Console from the First Steps window, click **Administrative Console**.
- To Launch Administrative Console from the browser, enter the following URL:
  - http://localhost:9090/admin

You should see the Administrative Console Login page, as shown in Figure 5-9.


IBM WebSphere Application Server Administrative Login - Netscape
File Edit View Go Bookmarks Tools Window Help
Search
Mail AIM Home Radio Netscape Search Bookmarks
IBM WebSphere Application Server ...
WebSphere App
Vers
nsole
IBM
Login
OK

http://neon:9091/admin/

informix@neon:/home/IDS9.40/demo/server

User ID:

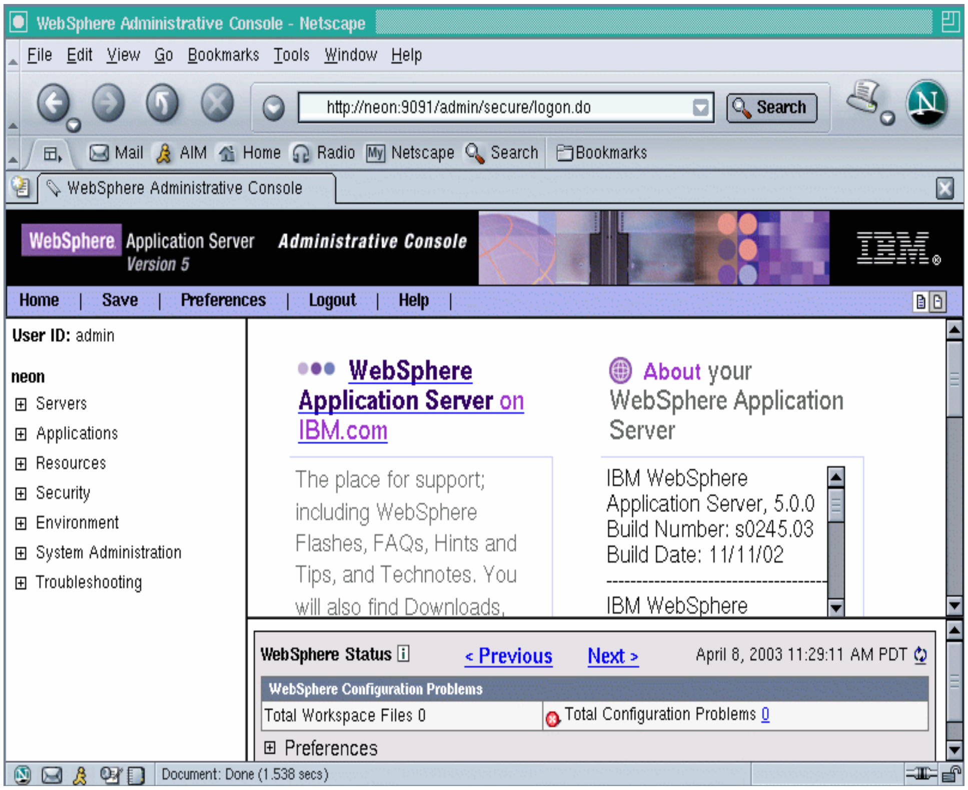
The User ID does not require a password, and does not need to be a User ID of a user in the local user registry. It is only used to track user-specific changes to configuration data. Security is NOT enabled

Transferring data from neon...

Figure 5-9 Administrative Login Page

WebSphere Application Server security is not enabled yet so we can enter any User ID to login to the Administrative Console. In our example, we used *admin* to login.

The page in Figure 5-10 will be displayed after logging in.


WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://neon:9091/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Administrative Console
WebSphere Application Server Administrative Console Version 5
IBM
Home | Save | Preferences | Logout | Help |
neon
Servers
Applications
Resources
Security
Environment
System Administration
Troubleshooting
WebSphere Application Server on IBM.com
About your WebSphere Application Server
< Previous
Next >
Preferences
Document: Done (1.538 secs)

User ID: admin

The place for support; including WebSphere Flashes, FAQs, Hints and Tips, and Technotes. You will also find Downloads,

IBM WebSphere Application Server, 5.0.0
Build Number: s0245.03
Build Date: 11/11/02
IBM WebSphere

WebSphere Status April 8, 2003 11:29:11 AM PDT

WebSphere Configuration Problems

Total Workspace Files 0 Total Configuration Problems 0

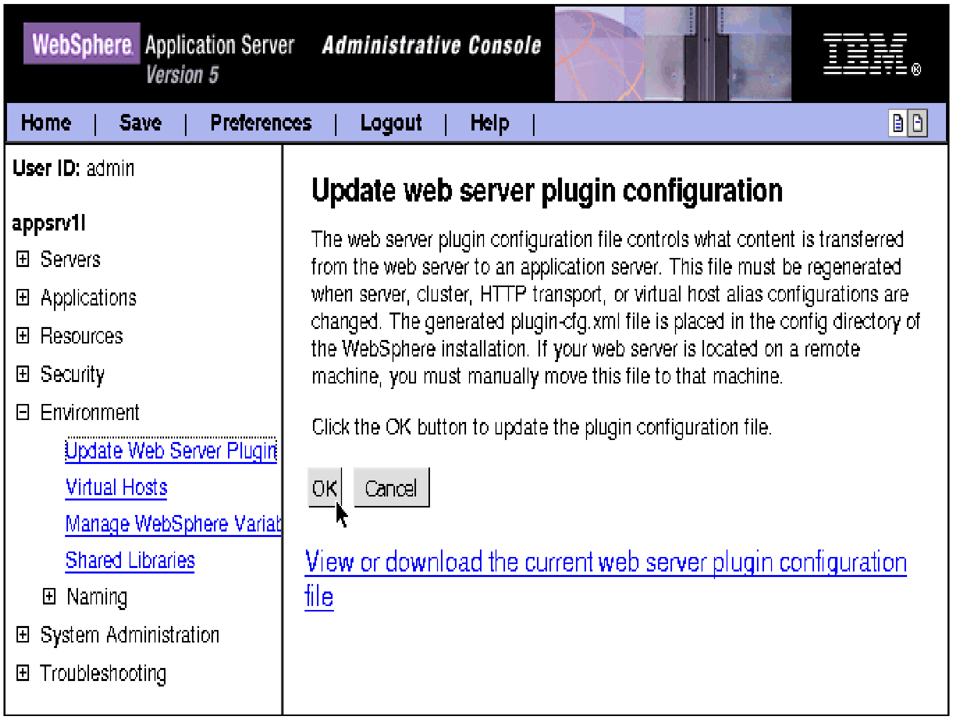
Figure 5-10 Administrative Console

Note: After installing WebSphere Application Server, you must update the Web Server Plug-in configuration. Updating Web Server Plug-in is described in the Redpaper, *IBM WebSphere V5.0 for Linux, Implementation and Deployment Guide WebSphere Handbook Series,* REDP3601. We have extracted the *Updating the Web server plug-in configuration* section from that Redpaper and have included it in the next section below.

Updating the Web server plug-in configuration

Changing certain WebSphere configuration properties makes it necessary to regenerate the Web server plug-in configuration. To accomplish this, do one of the following:

- From the Administrative Console, select **Environment -> Update Web Server Plugin** in the navigation tree, then click **OK**, as shown in Figure 5-11.



Update web server plugin configuration

The web server plugin configuration file controls what content is transferred from the web server to an application server. This file must be regenerated when server, cluster, HTTP transport, or virtual host alias configurations are changed. The generated plugin-cfg.xml file is placed in the config directory of the WebSphere installation. If your web server is located on a remote machine, you must manually move this file to that machine.

Click the OK button to update the plugin configuration file.

View or download the current web server plugin configuration file

Figure 5-11 Regenerating the Web server plug-in configuration from the console

- Use the script GenPluginCfg.sh script from the command line:

```
<WAS_HOME>/bin/GenPluginCfg.sh
```

5.2 Configuring WAS for use with IDS

After installing WebSphere Application Server 5.0 and IDS, we need to configure Informix as one of the JDBC Providers in WebSphere Application Server. Under Informix JDBC Provider, required data sources should be configured. Applications deployed in WebSphere Application Server can then connect to IDS, using the configured data sources / database URL.

WebSphere Application Server supports two Informix JDBC Providers, Informix JDBC Driver, and Informix JDBC Driver (XA) when there is a need to handle distributed transactions.

5.2.1 Configuring the Informix JDBC Provider

Here are the steps to configure the Informix JDBC Provider:

1. Start the WebSphere Application Server.
2. Open the Administrative Console as described in the section “Application server up and running” on page 112.
3. Click the **Resources -> JDBC Providers** link.
4. From the JDBC Providers frame, make sure the scope is set to Node.
5. Click the **New** button to create a new JDBC Provider.
6. From the New JDBC Provider frame, use the drop-down list box to choose the JDBC Provider named **Informix JDBC Driver (XA)**.

Note: For performing distributed transactions, use *Informix JDBC Driver (XA)*, otherwise use *Informix JDBC Driver.*

7. Click the **Apply** button to see the settings page of your JDBC Provider. Enter the following properties:

   Name:

   Informix JDBC Driver (XA)

   Classpath:

   <Informix JDBC Installed Directory>/lib/ifxjdbc.jar

   <Informix JDBC Installed Directory>/lib/ifxjdbcx.jar

   In our Demo: **/home/jdbc/lib/ifxjdbc.jar**

   /home/jdbc/lib/ifxjdbcx.jar

   Implementation Classname:

   com.informix.jdbcx.IfxXADataSource

8. Click the **Apply** button to set the changes
9. Click the **OK** button to complete the update

Note: Find the newly configured Informix JDBC Provider under the list of JDBC Providers.

10.. Click **Save** to save the configuration changes to the master repository.

5.2.2 Verifying the configuration

In Chapter 4, "Installing and configuring WebSphere Studio V5" on page 85, we described how to create a simple application using a database Web page to query the stores_demo data on IDS, and display the results on a browser. We used WSAD to create this sample application. Here we use the same application and deploy it on WAS. By doing this, we can verify the installation and configuration of WAS for IDS on the Linux server.

The last step described for WSAD was how to export a J2EE project in the form of an **EAR** file. Locate the ITSOStoresDBPages.ear file created and follow the steps in the next section to deploy it on the Application Server.

Deploy the ITSOStoresDBPages application in WAS

Here are the steps to deploy the ITSOStoresDBPages application in WAS:

1. Copy ITSOStoresDBPages.ear file into <WAS Installed Directory>\installableApps directory of the Application Server. We installed WebSphere Application Server on Linux. So, we copied this file to the /home/WAS_ENT/installableApps, using FTP. Now the application is ready to deploy to the Application Server.
2. Open the Administrative Console of WebSphere using the following URL:

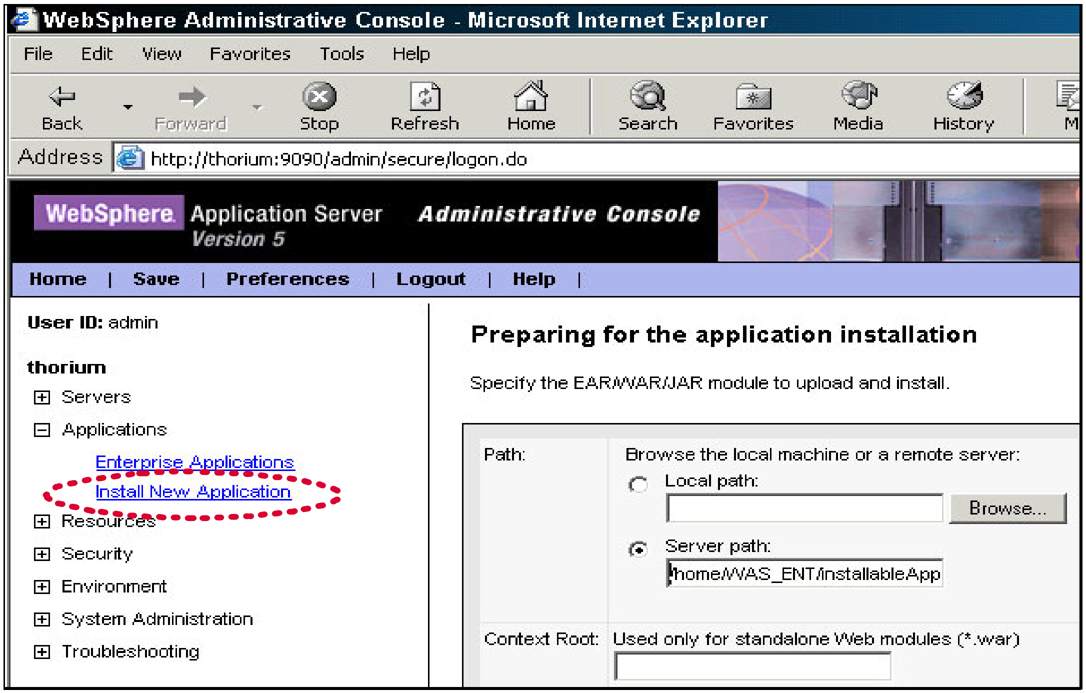
```
http://<server IP Address>:9090/admin
```

Verify that the server is running. For more information on starting and stopping the server, refer to the section "Application server up and running" on page 112. We can perform remote administration of the Application Server, so we accessed the Administrative Console from Windows 2000 client using the above-mentioned URL.

3. Enter **User ID** as *admin.* Click the **OK** button.

4. Click **Applications -> Install New Application** as shown in Figure 5-12. In the right pane, the page, **Preparing for the application installation**, will be displayed.

5. Select **Path** as *Server Path* and enter the path of the application as *<Application Server Server Installed Directory>\installableApps\ITSOStoresDBPages.ear.* We entered */home/WAS_ENT/installableApps/ITSOStoresDBPages.ear*. Click **Next** and the contents of the right pane will be changed.


WebSphere Administrative Console - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Forward
Stop
Refresh
Home
Search
Favorites
Media
History
M
Address
WebSphere Application Server Version 5
Administrative Console
Home | Save | Preferences | Logout | Help |
Browse...

Address: http://thorium:9090/admin/secure/logon.do

User ID: admin

thorium

⊞ Servers
⊟ Applications
  Enterprise Applications
  Install New Application
⊞ Resources
⊞ Security
⊞ Environment
⊞ System Administration
⊞ Troubleshooting

Preparing for the application installation

Specify the EAR/WAR/JAR module to upload and install.

Path: Browse the local machine or a remote server:

○ Local path:

● Server path: /home/WAS_ENT/installableApp

Context Root: Used only for standalone Web modules (*.war)

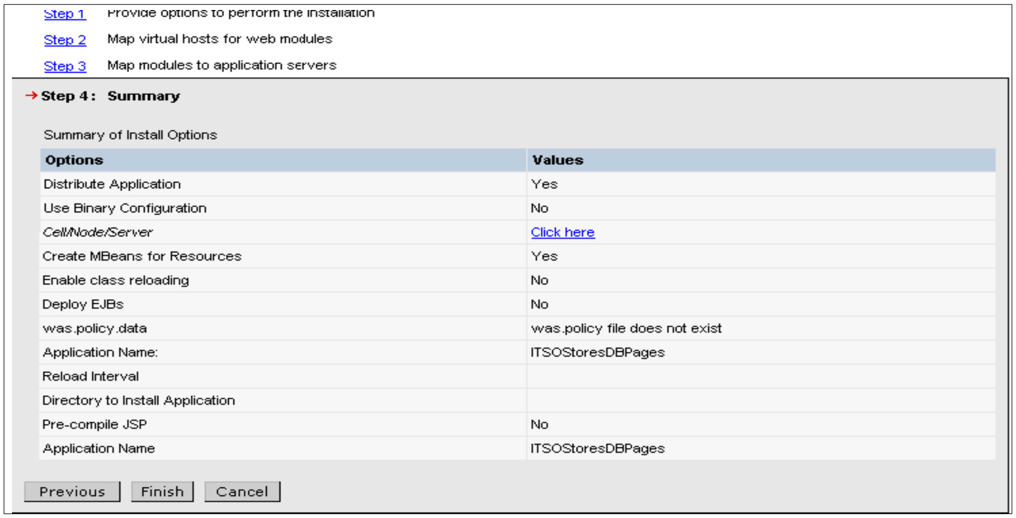
Figure 5-12 WebSphere Administrative Console

6. Click the **Next** button repeatedly, until you reach the **Finish** button. The default settings are fine for this application. Before clicking **Finish**, be sure to verify your summary page with Figure 5-13.

7. Click the **Finish** button. Wait for few minutes, then you will get the page shown in Figure 5-14.

8. Click the link **Save to Master Configuration** as shown in Figure 5-14.

9. Click **Save** to save all the information into the master configuration. We have now successfully deployed the **ITSOStoresDBPages** application on server. Now we need to start the application and run it.



Step 1 Provide options to perform the installation

Step 2 Map virtual hosts for web modules

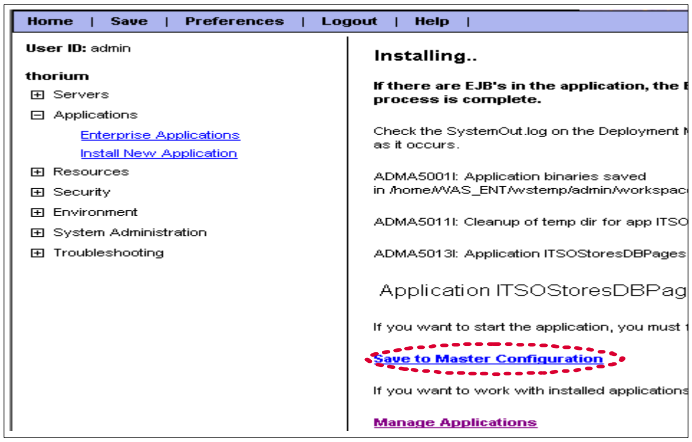
Step 3 Map modules to application servers

→ Step 4 : Summary

Summary of Install Options

Options	Values
Distribute Application	Yes
Use Binary Configuration	No
Cell/Node/Server	Click here
Create MBeans for Resources	Yes
Enable class reloading	No
Deploy EJBs	No
was.policy.data	was.policy file does not exist
Application Name:	ITSOStoresDBPages
Reload Interval	
Directory to Install Application	
Pre-compile JSP	No
Application Name	ITSOStoresDBPages

Figure 5-13 Summary page



User ID: admin

thorium

⊞ Servers

⊟ Applications

Enterprise Applications

Install New Application

⊞ Resources

⊞ Security

⊞ Environment

⊞ System Administration

⊞ Troubleshooting

Installing..

If there are EJB's in the application, the E process is complete.

Check the SystemOut.log on the Deployment M as it occurs.

ADMA5001I: Application binaries saved in /home/WAS_ENT/wstemp/admin/workspac

ADMA5011I: Cleanup of temp dir for app ITSO

ADMA5013I: Application ITSOStoresDBPages

Application ITSOStoresDBPag

If you want to start the application, you must t

Save to Master Configuration

If you want to work with installed applications

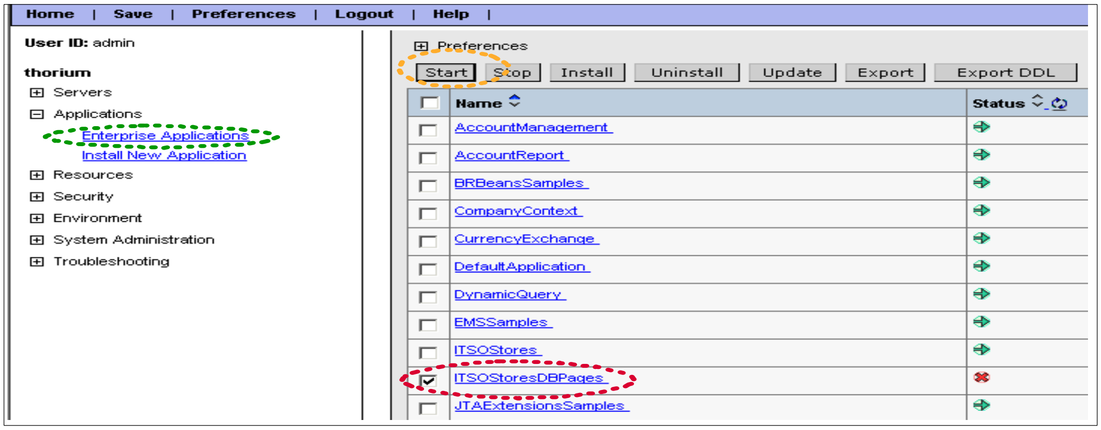
Manage Applications

Figure 5-14 Save to Master Configuration

Starting the application:

10.Click **Applications -> Enterprise Applications** as shown in Figure 5-15. Find the list of applications in the right pane.

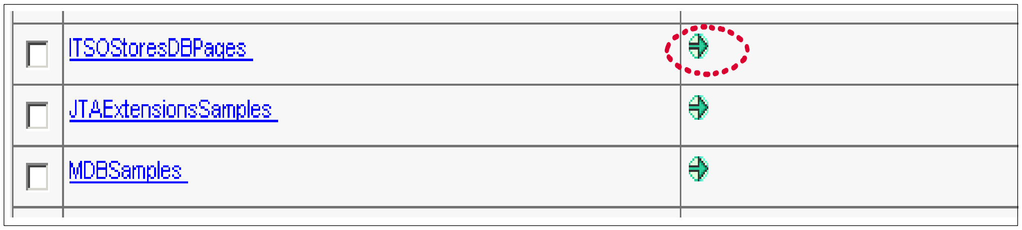
11.Select the application **ITSOStoresDBPages** as shown in. If you do not see the application, just scroll down the list of applications until you find it.



□	Name	Status
□	AccountManagement	➡
□	AccountReport	➡
□	BRBeansSamples	➡
□	CompanyContext	➡
□	CurrencyExchange	➡
□	DefaultApplication	➡
□	DynamicQuery	➡
□	EMSSamples	➡
□	ITSOStores	➡
☑	ITSOStoresDBPages	✖
□	JTAExtensionsSamples	➡

Figure 5-15 Starting the application

12. Click the **Start** button and the application will be started. The status of the application ITSOStoresDBPages is displayed as a green arrow mark, as shown in Figure 5-16. It indicates that we have successfully started the application. The only remaining task is to run the application.



□	ITSOStoresDBPages	➡
□	JTAExtensionsSamples	➡
□	MDBSamples	➡

Figure 5-16 Status of application

Running the application:

13. Open the browser and enter the following URL:

```
http://<I P Address of Server>:9080/ITSOStoresDBPagesWeb/itso_customerInputForm.html
```

An Input form will be opened:

14. Enter the **customer_num** as *102.* The Customer details will be displayed in **Select Result View**.

This completes the deployment of our simple WSAD developed application in the WebSphere Application Server. In subsequent chapters we will develop a more complex application, which also demonstrates deployment in a distributed environment.

5.3 WebSphere Administration Repository

Previous versions of WebSphere Application Server used a relational database to store the administration repository data. That meant that there was a prerequisite and dependency on having a relational database. However, WebSphere Application Server 5.0 uses XML files for this purpose, thus eliminating the dependency on a relational database.

The repository documents are stored in a directory tree starting at the *configuration* directory under the product installation root. At the top of the hierarchy is the cells directory, which contains a subdirectory for each cell. The names of the cell subdirectories match the names of the cells. For example, a cell named *mycell* stores its configuration documents in the directory, cells/mycell. In Application Server 5.0, there is always a single cell.

Each cell subdirectory contains the following files and subdirectories:

The *cell.xml* file, which provides configuration data for the cell, is located here.

Files such as *security.xml, virtualhosts.xml, resources.xml,* and *variables.xml*, provide configuration data that applies across every node in the cell.

The clusters subdirectory holds a subdirectory for each cluster defined in the cell. The names of the subdirectories under clusters match the names of the clusters. Each cluster subdirectory holds a *cluster.xml* file, which provides configuration data specifically for that cluster.

The applications subdirectory holds a subdirectory for each application deployed in the cell, and the nodes subdirectory holds a subdirectory for each node in the cell. The names of the subdirectories under nodes match the names of the nodes.

Each node subdirectory contains files such as *variables.xml* and *resources.xml*, which provide configuration data that applies across the node. Recall that these files have the same name as those in the containing cell's directory. The configurations specified in these node-level documents override the configurations specified in cell-level documents that have the same name. For example, if a particular variable is in both cell- and node-level *variables.xml* files, all servers on the node use the variable definition in the node-level document and ignore the definition in the cell-level document.

Each node subdirectory also contains a *serverindex.xml* file that stores the definitions of all ports used by servers on that node. Keeping this information in one document makes it easier to find any port conflicts between servers on a node.

Each node directory contains a subdirectory for each server defined on the node. The names of the subdirectories match the names of the servers. This directory always contains a *server.xml* file, which provides configuration data specific to that server. There might also be *security.xml*, *resources.xml* and *variables.xml* files, which provide configuration data that applies only to the server and overrides the configurations specified in the containing cell and node documents that have the same name.

The names of the applications subdirectories match the names of the deployed applications. The original EAR file for the application is stored in each application subdirectory. Also, under the application subdirectory is a deployments directory. You can deploy one application multiple times with different bindings for each deployment.

Each deployment subdirectory contains a *deployment.xml* file that contains configuration data about the application deployment. Each deployment subdirectory also contains a META-INF subdirectory that contains a J2EE application deployment descriptor file along with IBM deployment extensions and bindings files. Deployed application directories also have subdirectories for all WAR files and entity bean JAR files in the application. Only the metadata for the application is stored in the deployment subdirectories. The binary code for the application is stored in the location that the administrator specified when installing the application.

Note: Refer to Figure 5-17 for the location of some of the repository XML files. The directories are highlighted in blue and the files are highlighted in green. For a description of each repository XML file, and recommendations for editing those files, refer to the information at the following URL:

`http://publib7b.boulder.ibm.com/wasinfo1/en/info/aes/ae/rrun_rconfdoc_descriptions.html`

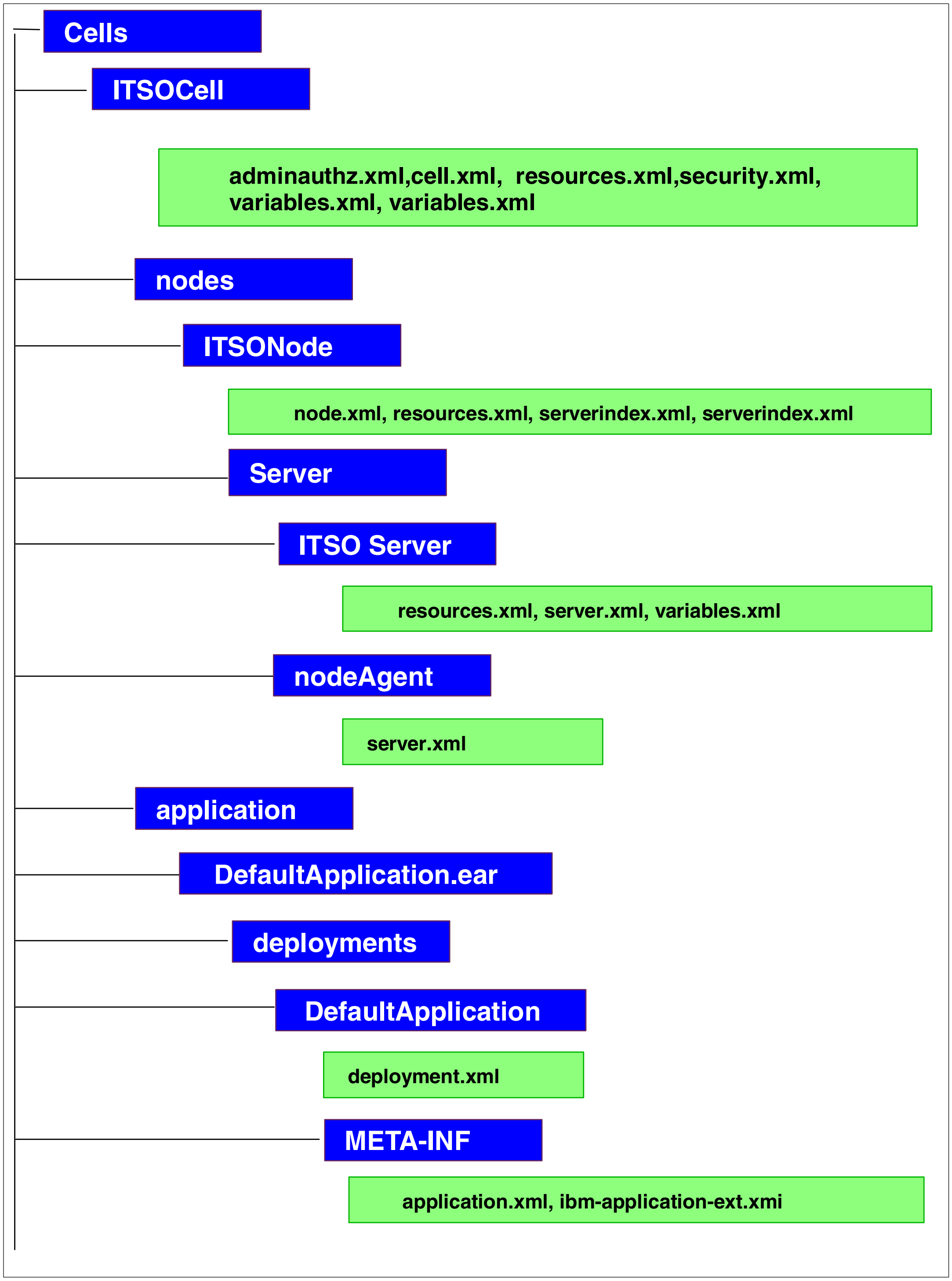

Cells
ITSOCell
adminauthz.xml,cell.xml, resources.xml,security.xml, variables.xml, variables.xml
nodes
ITSONode
node.xml, resources.xml, serverindex.xml, serverindex.xml
Server
ITSO Server
resources.xml, server.xml, variables.xml
nodeAgent
server.xml
application
DefaultApplication.ear
deployments
DefaultApplication
deployment.xml
META-INF
application.xml, ibm-application-ext.xmi

Figure 5-17 Administration repository files and their directory structure

6

Integrating IDS and WebSphere

The real power and benefit of Informix Dynamic Server and WebSphere significantly increases when you integrate the two together. We will demonstrate the integration in Chapter 7, “Working with IDS and WebSphere” on page 143, when we develop a sample application using IDS with WebSphere Studio, and deploy it to the WebSphere Application Server. In the current chapter we provide some of the background information and steps necessary to begin that integration.

The first step in this integration is to simply enable WebSphere to communicate with IDS. In preparation for that, we cover the following topics:

- **Scenarios for deploying WebSphere and IDS:** This topic describes the different environment configurations that can be used.
- **Systems architecture:** This topic gives an overview of the architecture we used when we configured IDS and WebSphere for our sample applications.
- **Making the connection:** This topic discusses connecting with IDS using a data source configuration on WebSphere Studio Application Developer.

6.1 Scenarios for deploying IDS and WebSphere

WebSphere is an integrated environment that is fully J2EE compliant. In Chapter 3, “WebSphere V5: An overview” on page 63, we showed the conceptual n-tier model of J2EE and described the benefits. In this section we present considerations and scenarios for deploying WebSphere and IDS, using the J2EE model. We discuss topics regarding scalability considerations, and present guidelines that can be followed during your implementation. We also describe some different models that may be considered when deploying a e-business system.

Scalability considerations

Understanding the scalability of the components of your e-business infrastructure, as well as applying appropriate techniques, can greatly improve availability and performance. Scaling techniques are especially important in multi-tier architectures. They enable you to evaluate components associated with dispatchers or edge servers, Web presentation servers, Web application servers, and data and transaction servers.

You can use the following high-level steps to classify your Web site and identify scaling techniques that are applicable to your environment:

- **Understand the application environment:** For existing environments, the first step is to identify all components and understand how they relate. The most important task is to understand the requirements and flow of the existing application(s) and what can or cannot be altered. The application is key to the scalability of any infrastructure, so a detailed understanding is mandatory to scale effectively. The transaction type and volume among the logical tiers should be analyzed closely so the correct resources are applied on each one of them. For example, for online banking, most of the latency typically occurs in the database server, whereas the application server typically experiences the greatest latency for online shopping and trading sites.
- **Categorize your workload:** Knowing your workload pattern (publish/subscribe or customer self-service, for example) determines where to focus your scalability efforts, and which scaling techniques to apply. For example, a customer self-service site, such as an online bank, needs to focus on transaction performance and the scalability of databases that contain customer information used across sessions. These considerations would not typically be significant for a publish/subscribe site. At this stage it is important to categorize your e-business site. Typically they fall under known classification types (online shopping, customer self-service, business-to-business, etc) and they tend to follow particular workload patterns.
- **Determine the components most impacted:** This step involves mapping the most important site characteristics to each component. Once again, from a scalability viewpoint, the key components of the infrastructure are the dispatchers or edge servers, the Web application servers, security services, transaction and data servers, and the network.
- **Select the scaling techniques to apply:** When the information gathering is as complete as it can be, it is time to consider matching scaling techniques to components. There are some typical guidelines that can be applied on the specific components of the multi-tier model, such as use faster servers, aggregate user data, and create a pool of pre-established connections.
- **Apply the techniques:** Do so in a test environment first, to evaluate the performance and scalability impacts to a component, as well as to determine how each change affects the surrounding components and the end-to-end infrastructure. You do not want a situation where improvements in one component result in an increased load on another component.

- **Reevaluate:** As with all performance related work, tuning will be required. This is a task that involves frequent reevaluation of what was implemented on previous stages. The goals are to eliminate bottlenecks, scale to a manageable status those that cannot be eliminated, and work around those that cannot be scaled.

6.1.1 Implementation scenarios

While a variety of factors come into play when considering the appropriate model for a WebSphere deployment, the primary factors include security, performance, throughput, availability, maintainability, and session state.

The emphasis of our server model scenarios will be on how to apply different techniques and component associations to provide scalability, load balancing, and failover.

Single server model

This is the simplest scenario that we can have when we deploy WebSphere and IDS. We compact all logical tiers on one single server. See Figure 6-1.

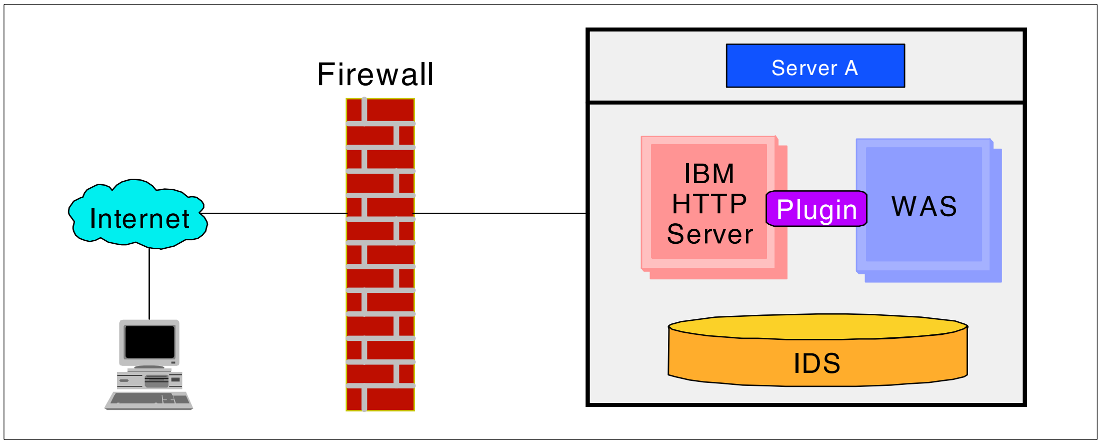


Figure 6-1 Single server model

This model requires fewer physical resources, since it utilizes only one server (for example a Linux server) to host the Web server, the application server, and the database server. This scenario is not normally used or recommended on a production system, but it is a good configuration for development and test purposes.

Here are some reasons to use a single server model:

- **Maintainability:** It is easy to install, administer, and maintain. This configuration is most suitable as a start-up configuration to evaluate and test the basic features of WebSphere and related components. The installation is automated by tools supplied with the WebSphere distribution.
- **When performance, security and availability are not critical goals:** This may be the case for development, testing, and some intranet environments. We are limited to the resources of a single server, which are shared by all components.
- **Low cost:** This model is economical.

Note: This is the deployment model used in our demo applications. Even though this configuration requires only one server, we recommend that it have plenty of memory and disk space. For instance, the server should have at least 512 MB.

Two-server model

For this scenario we use two servers to deploy our J2EE system. There are different configurations that can be selected, depending on the business needs. You can choose to isolate either the application server or the database server.

Here we illustrate one of the possible configurations, where we keep the database server on a dedicated server. By doing this we can perform specific tuning configuration, such as focusing on the database activities in a high data volume environment. If the system requires significant processing of business logic processing, and less database activity, you would most likely isolate the application server on a dedicated server instead.

Installing the database on a different server, in a two-server configuration model as depicted in Figure 6-2, represents a good practice with several advantages. These servers were used:

- **Server A:** IBM HTTP Server, WebSphere Application Server
- **Server B:** IBM Informix Dynamic Server

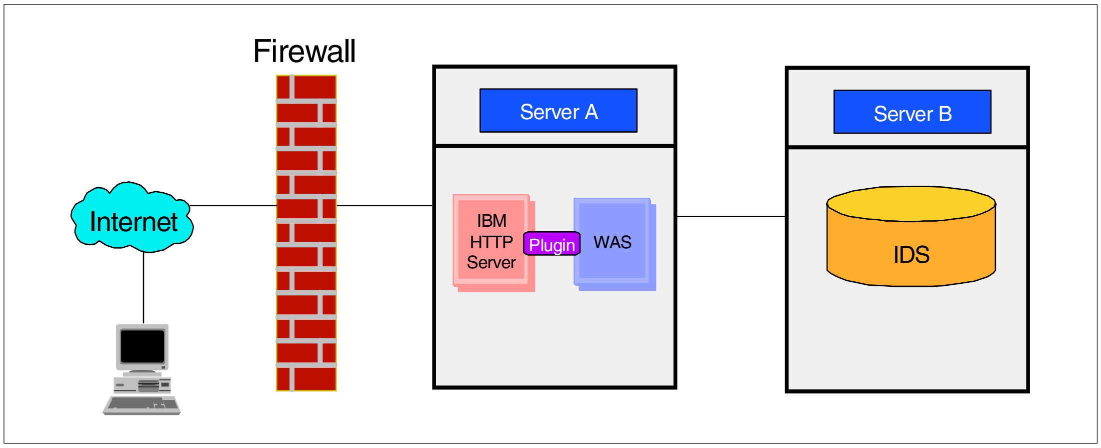


Figure 6-2 Two-server model with dedicated database server

Some of the advantages for separating the database servers are as follows:

- **Performance:** There is less competition for resources. If both the database and application server are placed on the same server, then under high load there are two processes. The application server and the database server compete for increasingly scarce resources, the CPU and the memory. In general, we can expect significantly better performance by separating the application server from the database server.
- **Performance:** Differentiated tuning is possible. By separating the servers, we can independently tune them to achieve optimal performance. The database server is typically sized and tuned for database performance, which may differ from the optimal configuration for the application server. On many UNIX servers, installing the database involves modification of the OS kernel. This database-specific tuning is often detrimental to the performance of application servers located on the same server.
- **Availability:** It allows use of already established highly available database servers. Many organizations have invested in high-availability solutions for their database servers, reducing the possibility of the server being a single point of failure in a system. If not, having the servers separated makes that task much easier.
- **Maintainability:** Individual components can be reconfigured, or even replaced, without affecting the installation of the other components.

Three-server model

This configuration consists of the Web server, application server, and database server installed in three different physical servers. Even though this scenario requires more hardware resources and administration work, it also provides more scalability, performance, and security than the other models. By using this model, you can fine-tune the servers for the particular tasks on each server. See Figure 6-3.

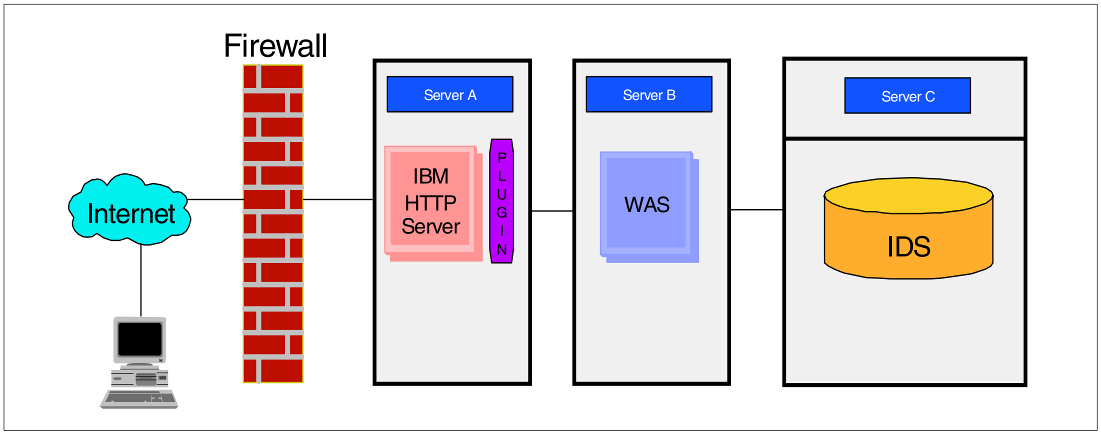


Figure 6-3 Three-server model

There many other configurations that can be considered. We have shown three typical configurations that are fairly basic and common ones. For more detailed information on the different deployment options and scalability, see the redbook entitled *IBM WebSphere V4.0 Advanced Edition Scalability*, SG24-6192.

6.2 High availability considerations

In the previous section we focused on issues of performance and scalability, demonstrating models that can be used to take best advantage of the winning combination of WebSphere and IDS. In other chapters of this book we discuss other technologies such as JSPs and Java servlets, as well as more advanced topics such as Enterprise Java Beans (EJBs) and Web services.

Since most Java enterprise developments will be deployed in a real world application server scenario, typically driving critical e-business and on-demand applications, you should design for their scalability, but even more important, the high availability of such a system. This section discusses the topic of high availability and provides scenarios for its deployment.

A typical WebSphere e-business application uses a three-tier approach (see Figure 6-4): the client tier (for example, a Web browser), the J2EE application server tier (1-n WebSphere application servers), and a database tier for the applications legacy data plus transactional EJB persistence. In some cases the database tier also maintains the application server configuration information.

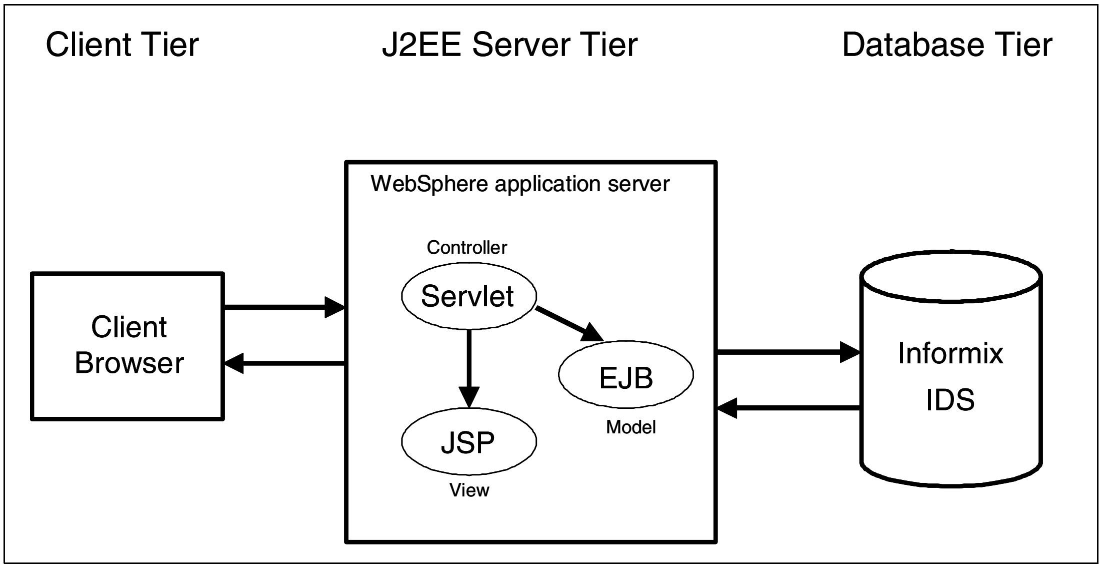


Figure 6-4 A typical three tier WebSphere architecture

On the WebSphere tier, high availability can be achieved by configuring multiple WebSphere application servers that handle the incoming client requests. Having multiple application servers also helps with the overall scalability and load balancing of such a system.

So how can we increase the availability for the underlying database system (IDS) which is a critical component for the overall reliability of most J2EE applications?

Before we discuss possible solutions to this problem we'll take a brief look at the information that could be stored in a WebSphere database.

What kind of data is stored in a J2EE database?

The data for a J2EE application, accessed through a WebSphere application server, can be defined in three categories:

- **Legacy data for the application:** This category includes customer records and order entries. Those data are typically provided by ERP applications such as SAP R/3, Peoplesoft, and Baan. In large scale applications the legacy database is normally separated from the WebSphere database.
- **Persistent storage for WebSphere EJBs:** Between each invocation of an EJB the bean needs to store its contents in the database to assure that it is available to the client, even if one of the application servers fail. Since an EJB could contain, for example, critical order entry records for an e-business application, one has to make sure that the persistent EJB storage is always accessible.
- **Method of storing configuration information:** WebSphere application server (WAS) 4.0.5 uses relational databases (including Informix Dynamic Server) for its configuration information. However, starting with version 5.0 of WAS, the configuration data is stored locally in XML files.

Since the first case is somewhat independent from the WebSphere specific databases, we'll focus on the high availability of the persistent EJB storage and the WAS (prior to version 5) configuration database.

6.2.1 IDS high availability and WebSphere

In addition of being one of the leading high-end OLTP database systems in the market, and one that already addresses most of the WebSphere scalability requirements, IDS also offers one of the leading high availability technologies: High-Availability Data Replication (HDR). HDR is a built-in, no cost option in IDS 7 and IDS 9 that allows application transparent replication of critical transactional data to a failover (secondary) server.

Why is HDR better than simple disk mirroring? Mirrored disks can protect you against a local disk failure and are a very good choice to increase the availability of a local IDS instance but they can't protect you against a catastrophic failure of a computer on which the IDS server is running — as can HDR. Using such a replication technology also allows the setup of a remote database replica hundreds of miles away from the primary database.

The next sections gives an introduction into the topic of HDR and how to configure a JDBC connection to utilize a replicated IDS database.

6.2.2 High availability with HDR

The IDS database server implements nearly transparent data replication of entire database servers. All the data that one database server manages is replicated to, and dynamically updated on, another database server, typically at a different geographical location. HDR provides a way to maintain a backup copy of the entire database server that applications can access quickly in the event of a catastrophic failure.

HDR server pairs are composed of a primary and a secondary server. The primary server is the default server. The secondary server is read-only, so update operations are not allowed. The secondary server in a HDR scenario can be used to off load read-only transactions from the primary server, if required.

Type of data replicated

HDR replicates data in dbspaces and sbspaces, but not in blobspaces, and it replicates all built-in and extended data types. User-defined types (UDTs) must be logged and reside in a single database server. Data types with out-of-row data are replicated if the data is stored in an sbspace or in a different table on the same database server. For data stored in an sbspace to be replicated, the sbspace must be logged. HDR does not replicate data stored in operating system files or persistent external files or memory objects associated with user-defined routines.

Replication modes supported

The primary database server sends the contents of the internal HDR buffer to the secondary database server either synchronously or asynchronously. The value of the ONCONFIG configuration parameter, DRINTERVAL, determines whether the database server uses synchronous or asynchronous updating. For more information on DRINTERVAL, see the chapter on configuration parameters in the IBM Informix IDS Administrator's Reference.

HDR failures, and actions taken in failure situations

An HDR failure is a loss of connection between the database servers in a replication pair. Any of the following situations might cause a data replication failure:

- Catastrophic failure (such as fire or earthquake) at the site of one of the database servers.
- Disruption of the networking cables that join the two database servers.
- Excessive delay in processing on one of the database servers.
- Disk failure on the secondary database server that is not resolved by a mirrored chunk.

Detection of an HDR failure

The database server interprets either of the following conditions as an HDR failure:

- A specified time-out value was exceeded.

  During normal HDR operation, a database server expects confirmation of communication from the other database server in the pair. Each database server in the pair has an ONCONFIG parameter, DRTIMEOUT, that specifies a number of seconds. If confirmation from the other database server in a pair does not return within the number of seconds that DRTIMEOUT specifies, the database server assumes that an HDR failure has occurred.

- A database server does not respond to the periodic messaging (pinging) attempts over the network.

  Each of the database servers sends a ping to the other database server in the pair when the number of seconds specified by the DRTIMEOUT parameter on that database server has passed. The database servers ping each other regardless of whether the primary database server sends any records to the secondary database server. If a database server does not respond to four ping attempts in a row, the other database server assumes that an HDR failure has occurred.

Actions to take if the primary server fails

If the primary database server fails, the secondary database server can behave in the following ways:

- The secondary database server can remain in logical-recovery mode. In other words, no action is taken. This would be the case if you expect the HDR connection to be restored very soon.
- Use manual switchover to change the database server mode. Manual switchover means that the administrator of the secondary database server changes the type of the secondary database server to standard. The secondary database server rolls back any open transactions and then comes into online mode as a standard database server, so that it can accept updates from client applications. A manual switchover doesn’t have to be manual since the primary failure also triggers an optional execution of any kind of shell script which could execute the necessary commands to change the secondary server mode. So a failover can be handled within a few seconds!

For a more detailed description of HDR, how to configure HDR and how to react in the different failure situations, please refer to the *IBM Informix IDS Administrator’s Guide*.

6.2.3 JDBC support for IDS and HDR

The IBM Informix JBDC driver supports connections to server pairs in an HDR replication scenario. In order to activate HDR support, one has to the set following secondary server properties in the connection URL:

- `INFORMIXSERVER_SECONDARY` = <secondary_server>;
- `PORTNO_SECONDARY` = <secondary_portnumber>;
- `IFXHOST_SECONDARY` = <secondary_hostmachine>;
- `ENABLE_HDRSWITCH` = true;

Example 6-1 shows a connection URL for an HDR server pair named hdr1 and hdr2.

Example 6-1 Example JDBC connection URL with HDR support

```
jdbc:informix-sqli://123.45.67.89:1533/testDB:INFORMIXSERVER=hdr1;IFXHOST=host1;PORTNO=1500
;user=rdtest;password=test;INFORMIXSERVER_SECONDARY=hdr2;IFXHOST_SECONDARY=host2;PORTNO_SEC
ONDARY=1600;ENABLE_HDRSWITCH=true;
```

When using a DataSource object, you can set and get the secondary server connection properties with setXXX() and getXXX() methods.

Using HDR with connection pooling

IBM Informix JDBC Driver implementation of connection pooling provides the ability to pool connections with database servers in an HDR pair:

The primary pool contains connections to the primary server in an HDR pair.
The secondary pool contains connections to the secondary server in an HDR pair.

You do not have to change application code to take advantage of connection pooling with HDR. Set the `IFMX_CPM_ENABLE_SWITCH_HDRPOOL` property to true to allow switching between the two pools. When switching is allowed, the Connection Pool Manager validates and activates the appropriate connection pool.

When the primary server fails, the Connection Pool Manager activates the secondary pool. When the secondary pool is active, the Connection Pool Manager validates the state of the pool to check if the primary server is running. If the primary server is running, the Connection Pool Manager switches new connections to the primary server and sets the active pool to the primary pool.

If `IFMX_CPM_ENABLE_SWITCH_HDRPOOL` is set to false, you can force switching to the other `connection pool by calling the activateHDRPool_Primary()` or `activateHDRPool_Secondary()` methods:

```
public void activateHDRPool_Primary(void) throws SQLException
public void activateHDRPool_Secondary(void) throws SQLException
```

The `activateHDRPool_Primary()` method switches the primary connection pool to be the active connection pool. The `activateHDRPool_Secondary()` method switches the secondary connection pool to be the active pool.

For more detailed information about the JDBC support for HDR, please refer to the IBM Informix JDBC Driver Programmer's guide.

Summary

The Informix Dynamic Server HDR capability provides a unique infrastructure that can enable WebSphere based applications to increase their availability. This is accomplished by a transparent replication mechanism. All IDS/WebSphere based J2EE applications that have the need for persistent and reliable EJB storage, or WAS 4.0.5 users that require a reliable, highly available configuration database, should consider using the HDR technology in their environment.

6.3 Systems architecture

In this section we show the architecture used in our installation. In the following chapters we will demonstrate the development and execution of some small sample applications, built in WebSphere Studio Application Developer and deployed in WebSphere Application Server, which are integrated with IDS.

Since our implementation is used primarily for testing and education purposes, we concentrated most of the software components in one single host server (the single-server model). WebSphere Studio Application Developer was installed on a separate server.

These are the products that we used:

- Linux SuSE 8.0: WebSphere Application Server 5.0, Informix Dynamic Server 9.40
- Windows 2000: WebSphere Studio Application Developer 5.0

The demo applications presented in this book were developed along with explanations of the technology used, so each application uses the appropriate Java components that are relevant to the explanations.

The illustration in Figure 6-5 shows the architecture of the configuration used for development of our demo applications. We first developed and tested the applications on WebSphere Studio, which was installed on a Windows client. WS Studio has its own embedded application and Web server, so if the application ran on our Windows client we could be confident it would run perfectly when deployed on the WS Application Server. We implemented the components and designed the sample applications to be developed in WS Studio, and to minimize the use of, and requirement for, detailed Java programming.

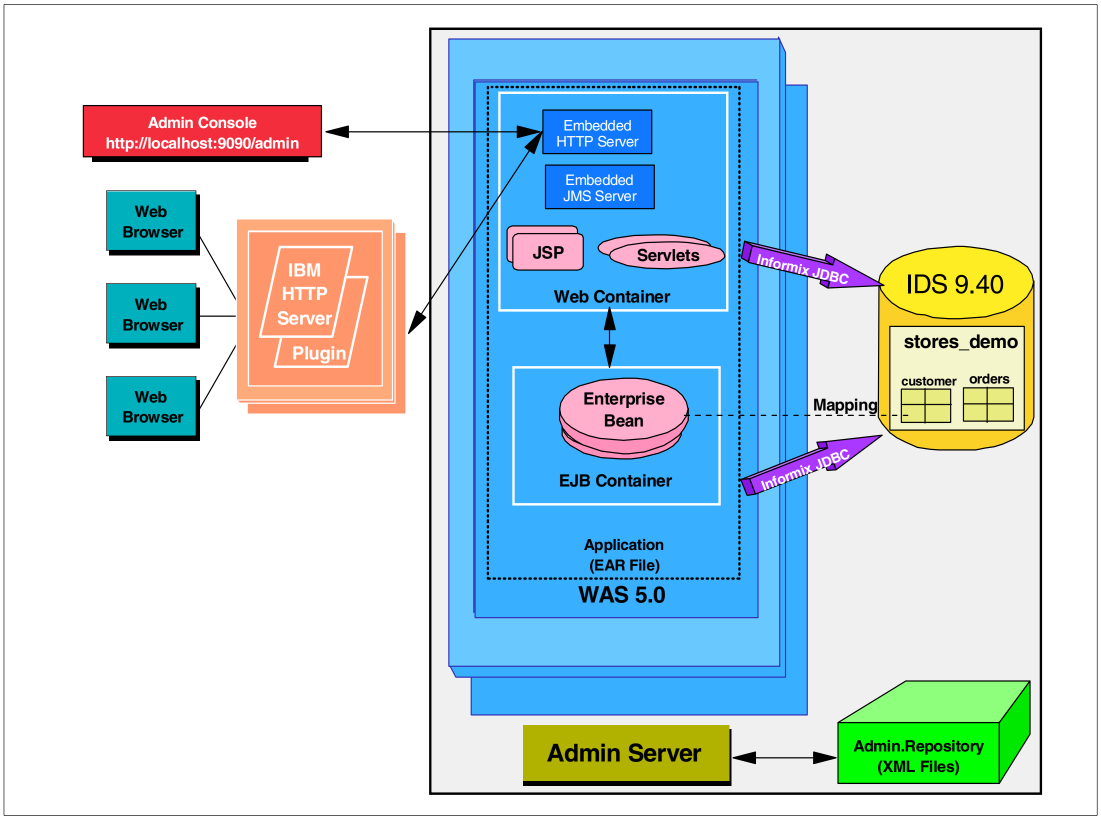


Figure 6-5 Systems architecture

6.4 Connecting IDS and WebSphere

To query an IDS database, we had to first establish a connection between the Informix database server and WebSphere. You can establish a connection by completing two actions:

1. Load the Informix JDBC Driver. The Informix JDBC Driver is based on Version 2.0 of the JDBC API.
2. Create a connection to either a database server, or a specific database, in one of the following ways:
   - Use a Data Source object.
   - Use the DriverManager.getConnection method.

6.4.1 Using a Data Source object

Using a Data Source object is preferable to using the DriverManager.getConnection method because a Data Source object is portable and makes the details if the underlying data source transparent to the application. The target data source implementation can also be modified, or the application can be redirected to a different server, without impacting the application code. A Data Source object can also provide support for connection pooling and distributed transactions. In addition, Enterprise Java Beans and J2EE require a Data Source object.

In JDBC 1.0 the only way to establish a database connection was by using the driver manager interface. This was expensive in terms of performance because a connection was created each time you needed to access the database from your program, resulting in substantial processing overhead. In the JDBC 2.0 Standard Extension API, an alternative means of handling database connections was introduced.

By using Data Source objects you have access to a pool of connections to a data source. Then, by using connection pooling you get the following advantages:

- It improves performance. Creating connections is expensive, a Data Source object creates a connection as soon as it is instantiated.
- It simplifies resource allocation. Resources are only allocated from the Data Source objects, and not by the code.
- It simplifies connection calls. To get a connection in JDBC 1.0, you had to call Class.forName() on the class name of the database driver, before making DriverManager calls.

Data Source objects work as follows:

- When a servlet or other client wants to use a connection, it looks up a Data Source object by name from a JNDI (Java Native Directory Interface) server.

Note: The Java Naming and Directory Interface (JNDI) is an application programming interface (API) that provides naming and directory functionality to applications written using the Java programming language. The JNDI architecture consists of an API and a service provider interface (SPI). Java applications use the JNDI API to access a variety of naming and directory services. The JNDI is included in the Java 2 SDK, v1.3 and later releases and includes three service providers for the following naming/directory services: Lightweight Directory Access Protocol (LDAP), Common Object Request Broker Architecture (CORBA) Common Object Services (COS) name service and Java Remote Method Invocation (RMI) Registry

- The Data Source object then returns a connection to the client.
- If the Data Source object has no more connections, it may ask the database manager for more connections (as long as it has not exceeded the maximum number of connections).
- When the client has finished with the connection, it is released.
- The Data Source object then returns the connection to the available pool. The next time the connection is needed, it is already created and can be reused without the overhead of recreating the connection.

In the following sections we show how to create and configure an Informix data source on both WebSphere Studio and WebSphere Application Server. This chapter is in preparation for the following ones, where we develop a sample application to demonstrate how IDS and WebSphere work well together. The sample application is built using WSAD and deployed on WAS, so they both need to have the resources created.

6.4.2 Configure Informix Data Source on WebSphere Studio

In this section we show, step by step, how to configure a data source connection to IDS 9.40 using WebSphere Studio.

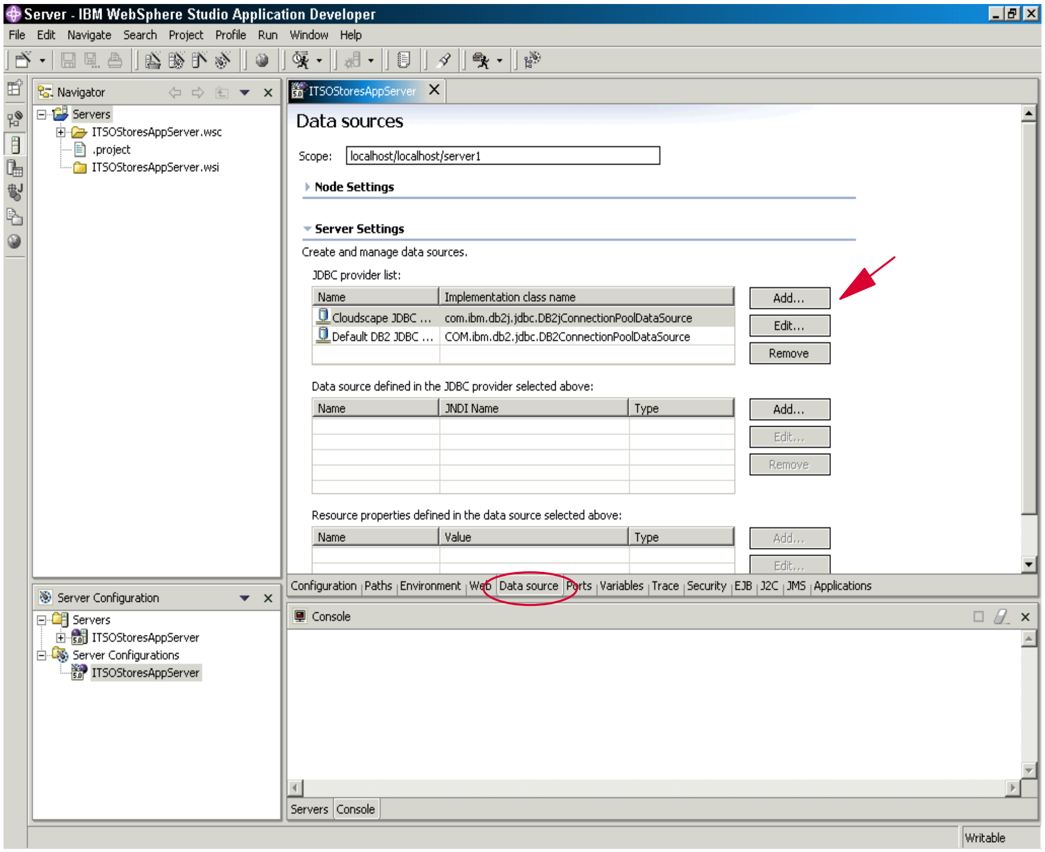
Overview steps for the configuration process

Here are the overview steps to configure an Informix data source on WebSphere Studio:

1. Locate the **ITSOStoresAppServer** server configuration in the Server Perspective.
2. Create a new JDBC provider using Informix JDBC driver.
3. Create a new JAAS authentication entry for the server configuration.
4. Create the data source using the Informix JDBC provider.

Detailed steps to configure an Informix Data Source on WSAD

1. Open WebSphere Studio and select your testing workspace.
2. Select the Server perspective (**Window -> Open Perspective -> Server**) and locate the server configuration called **ITSOStoresAppServer** that was created in 4.3, “A sample application using Database Web Pages” on page 88 (in step 32).
   You should find this in the **Server Configuration** window.
3. Next, double-click the **ITSOStoresAppServer** server configuration. A new window will open on the top right frame.
   a. Select the **Data Source** tab.
   b. Add a new JDBC provider. See Figure 6-6.



Scope: localhost/localhost/server1

Create and manage data sources.

JDBC provider list:

Name	Implementation class name
Cloudscape JDBC ...	com.ibm.db2j.jdbc.DB2jConnectionPoolDataSource
Default DB2 JDBC ...	COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource

Data source defined in the JDBC provider selected above:

Name	JNDI Name	Type

Resource properties defined in the data source selected above:

Name	Value	Type

Figure 6-6 Adding a new JDBC provider

c. Select **Informix** as the Database type.

d. Select **Informix JDBC Driver (XA)** as the JDBC provider type.

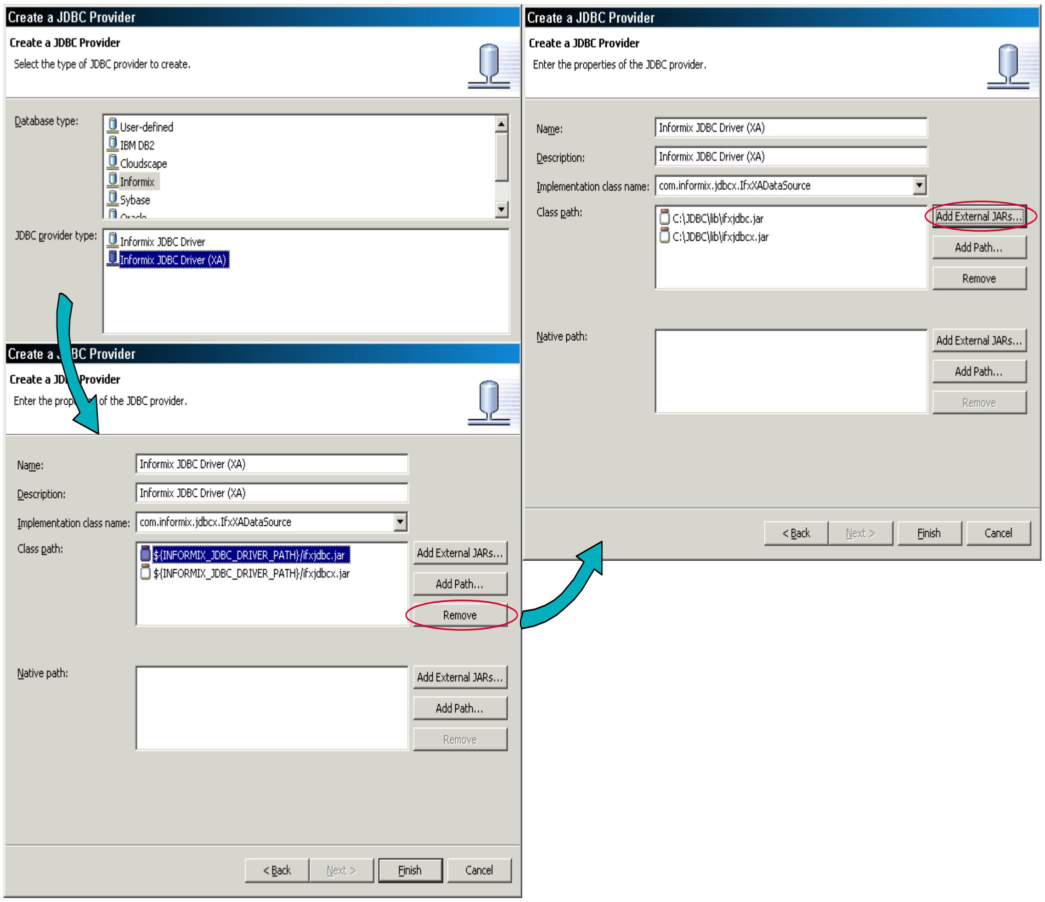
e. Select a name for the JDBC provider. In our case we named it Informix JDBC Driver (XA).

f. Remove the default jar files.

g. Add the jar file with the correct paths. You can browse through your computer and look for the ifxjdbc.jar and ifxjdbcx.jar in the <JDBC_DIR>/lib directory.

> **Note:** The Informix JDBC driver has been previously installed — see section 4.2, “Configuring WSAD for IDS” on page 87.

h. Click **Finish**. See Figure 6-7.



Create a JDBC Provider

Create a JDBC Provider

Select the type of JDBC provider to create.

Database type: User-defined, IBM DB2, Cloudscape, Informix, Sybase, Oracle

JDBC provider type: Informix JDBC Driver, Informix JDBC Driver (XA)

Create a JDBC Provider

Create a JDBC Provider

Enter the properties of the JDBC provider.

Name: Informix JDBC Driver (XA)

Description: Informix JDBC Driver (XA)

Implementation class name: com.informix.jdbcx.IfxXADataSource

Class path:
${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbc.jar
${INFORMIX_JDBC_DRIVER_PATH}/ifxjdbcx.jar

Native path:

Create a JDBC Provider

Create a JDBC Provider

Enter the properties of the JDBC provider.

Name: Informix JDBC Driver (XA)

Description: Informix JDBC Driver (XA)

Implementation class name: com.informix.jdbcx.IfxXADataSource

Class path:
C:\JDBC\lib\ifxjdbc.jar
C:\JDBC\lib\ifxjdbcx.jar

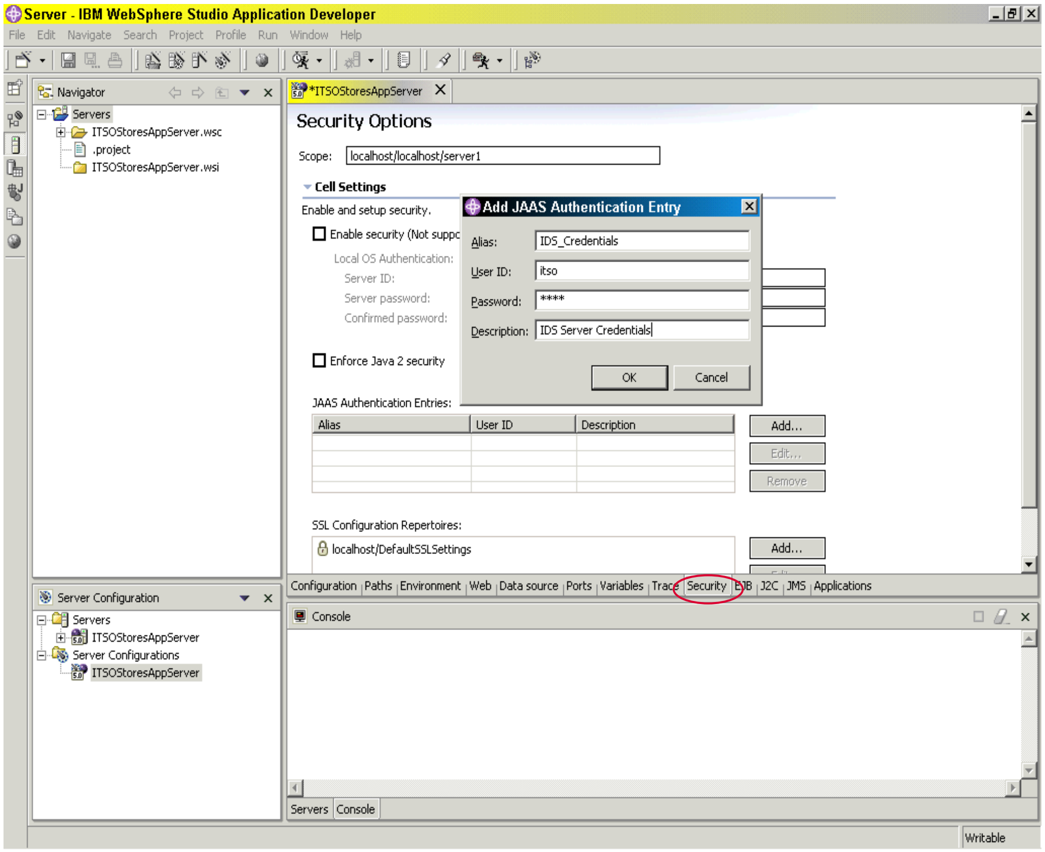
Native path:

Figure 6-7 Configuring JDBC provider

i. The last step in the **ITSOStoresAppServer** server configuration is to configure a user authentication mechanism called JAAS.

> **Note:** The Java Authentication and Authorization Service (JAAS) is a package that enables services to authenticate and enforce access controls upon users. It implements a Java version of the standard Pluggable Authentication Module (PAM) framework, and supports user-based authorization.

j. Click the security tab and add a new JAAS authentication entry. Select a name and the user and password that will be registered. In our case we called it `IDS_Credentials` and we registered the itso user with its appropriate password. Click **OK** and you will see a new entry in the JAAS list. See Figure 6-8.



Scope: localhost/localhost/server1

Alias: IDS_Credentials

User ID: itso

Password: ****

Description: IDS Server Credentials

Figure 6-8 Adding a new JAAS entry

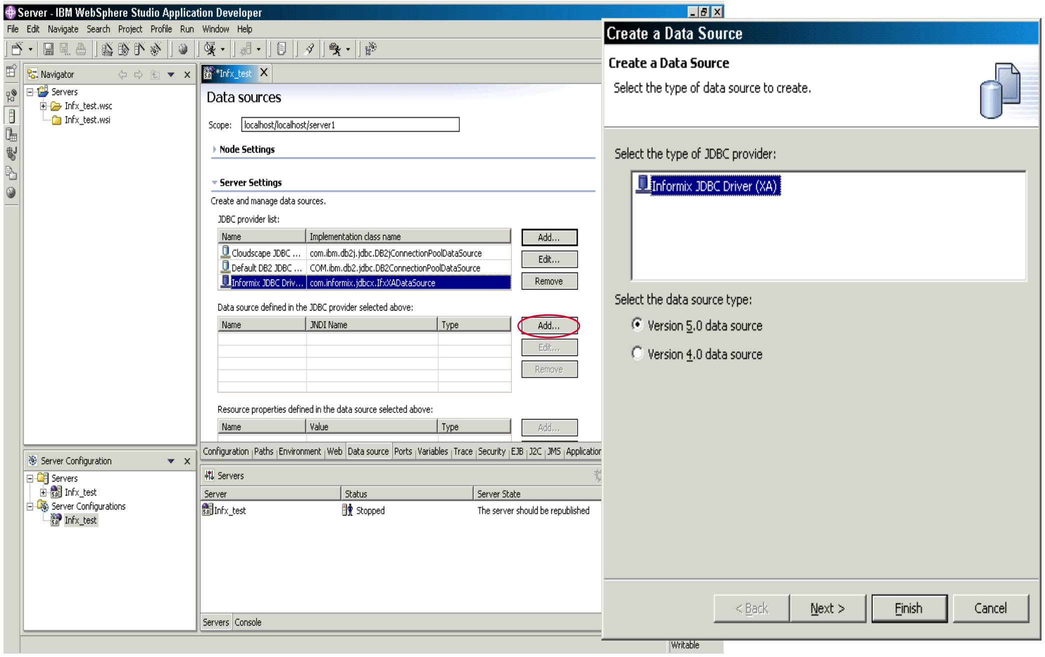
4. Now we need to create a new Data Source using the Informix JDBC provider previously created. Follow these steps:

   a. Select the **Data Source** tab again.

   b. Select the Informix JDBC provider in the JDBC provider list.

   c. Click the **Add** button in the Data Source list.

   d. Choose the Informix JDBC provider (XA) and use version 5.0 for the Data Source. See Figure 6-9.



Data sources

Scope: localhost/localhost/server1

Create and manage data sources.

JDBC provider list:

Name	Implementation class name
Cloudscape JDBC ...	com.ibm.db2j.jdbc.DB2jConnectionPoolDataSource
Default DB2 JDBC ...	COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
Informix JDBC Driv...	com.informix.jdbcx.IfxXADataSource

Data source defined in the JDBC provider selected above:

Name	JNDI Name	Type

Resource properties defined in the data source selected above:

Name	Value	Type

Server	Status	Server State
Infx_test	Stopped	The server should be republished

Create a Data Source

Create a Data Source

Select the type of data source to create.

Select the type of JDBC provider:

Informix JDBC Driver (XA)

Select the data source type:

● Version 5.0 data source

○ Version 4.0 data source

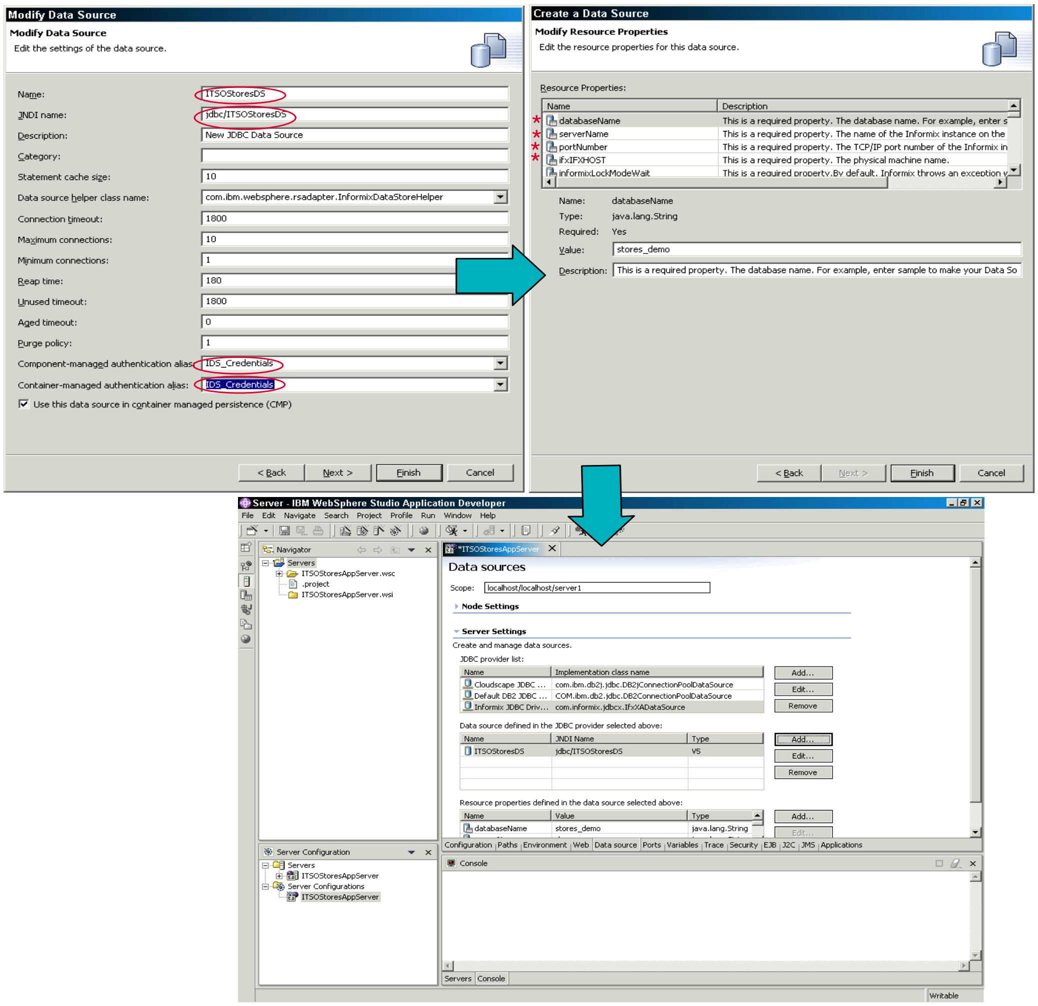
Figure 6-9 Adding new data source

e. In the Modify Data Source window, enter the name and the JNDI name for the Data Source. We used the names **ITSOStoresDS** and **jdbc/ITSOStoresDS**. Also specify the JAAS configuration (**IDS_Credentials**) in the Component-Managed Authentication alias and Container-Managed Authentication alias. Click **Next**.

f. Now we need to fill out the configuration parameters that are specific to the Informix JDBC driver (Create a Data Source window). There are four mandatory parameters that need to be specified. They are:

- **databaseName:** Database name (stores_demo)
- **serverName:** Informix server instance (demo_on)
- **portNumber:** TCP/IP port number used for the connection (1533)
- **ifxIFXHOST:** Host name or IP address (9.1.38.76)

g. Click the **Finish** button and you'll see a new entry in the Data Source list. See Figure 6-10.



Modify Data Source

Edit the settings of the data source.

Name:	ITSOStoresDS
JNDI name:	jdbc/ITSOStoresDS
Description:	New JDBC Data Source
Category:	
Statement cache size:	10
Data source helper class name:	com.ibm.websphere.rsadapter.InformixDataStoreHelper
Connection timeout:	1800
Maximum connections:	10
Minimum connections:	1
Reap time:	180
Unused timeout:	1800
Aged timeout:	0
Purge policy:	1
Component-managed authentication alias:	IDS_Credentials
Container-managed authentication alias:	IDS_Credentials

☑ Use this data source in container managed persistence (CMP)

Modify Resource Properties

Edit the resource properties for this data source.

Resource Properties:

Name	Description
* databaseName	This is a required property. The database name. For example, enter s
* serverName	This is a required property. The name of the Informix instance on the
* portNumber	This is a required property. The TCP/IP port number of the Informix in
* ifxIFXHOST	This is a required property. The physical machine name.
informixLockModeWait	This is a required property.By default, Informix throws an exception v

Name: databaseName

Type: java.lang.String

Required: Yes

Value: stores_demo

Description: This is a required property. The database name. For example, enter sample to make your Data So

Data sources

Scope: localhost/localhost/server1

Node Settings

Server Settings

Create and manage data sources.

JDBC provider list:

Name	Implementation class name
Cloudscape JDBC ...	com.ibm.db2j.jdbc.DB2jConnectionPoolDataSource
Default DB2 JDBC ...	COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource
Informix JDBC Driv...	com.informix.jdbcx.IfxXADataSource

Data source defined in the JDBC provider selected above:

Name	JNDI Name	Type
ITSOStoresDS	jdbc/ITSOStoresDS	V5

Resource properties defined in the data source selected above:

Name	Value	Type
databaseName	stores_demo	java.lang.String

Figure 6-10 Configuring the Informix data source

Now that an Informix data source was created, we are ready to create demo applications. This data source will be used in Chapter 7, “Working with IDS and WebSphere” on page 143, when a full application is developed in WebSphere Studio.

6.4.3 Configure Informix Data Source on WebSphere Application Server

After you create an application on WSAD, you need to deploy it on WAS. Usually the connection to the database server is performed through a data source, so in most of our demo applications, we use this method to connect to IDS. Therefore, before deploying the created application, we need to prepare the WAS environment. In this section we explain how you can create an Informix data source using the WAS Administrative Console.

Overview steps for the configuration process

Here are the overview steps to configure an Informix data source on WAS:

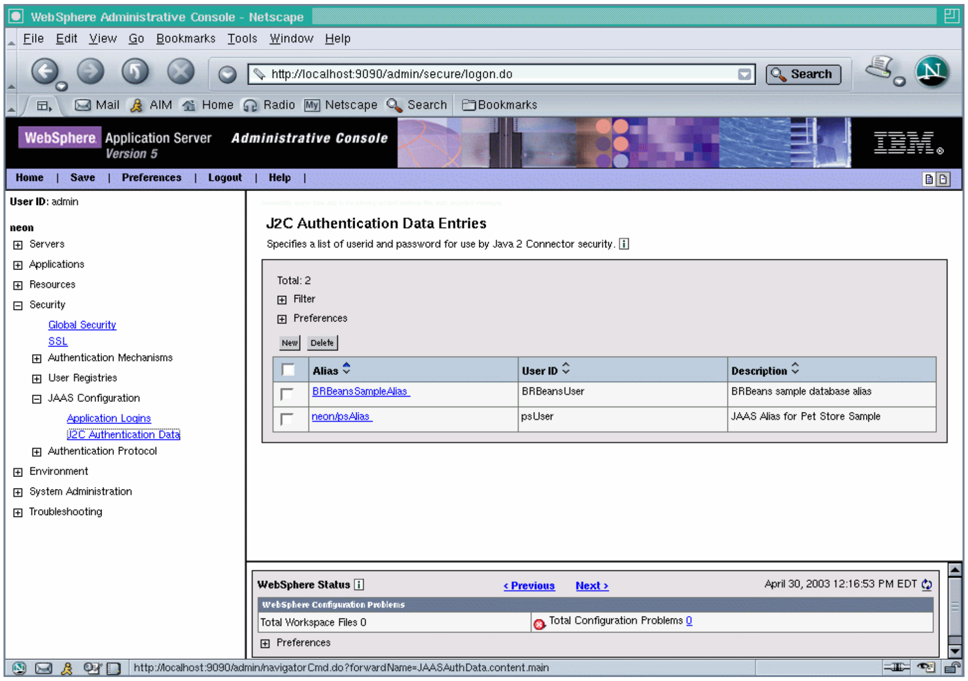
1. Log into the Administrative Console.
2. Create a new JAAS authentication entry on the security tab.
3. Use the Informix JDBC provider (configured earlier in chapter 5) to create a new data source.

Detailed steps to configure an Informix Data Source on WAS

1. Open a Web browser and log into the Administrative Console:

   http://localhost:9090/admin

2. Before creating the data source, we first need to create a new JAAS authentication entry. To do that, expand the security tab on the left frame and choose **J2C authentication Data.** Click the **New** button to add a new entry. See Figure 6-11.


WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:9090/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
neon
Servers
Applications
Resources
Security
Global Security
SSL
Authentication Mechanisms
User Registries
JAAS Configuration
Application Logins
J2C Authentication Data
Authentication Protocol
Environment
System Administration
Troubleshooting
New
Delete
< Previous
Next >
http://localhost:9090/admin/navigatorCmd.do?forwardName=JAASAuthData.content.main

J2C Authentication Data Entries

Specifies a list of userid and password for use by Java 2 Connector security.

Total: 2

⊞ Filter

⊞ Preferences

□	Alias	User ID	Description
□	BRBeansSampleAlias	BRBeansUser	BRBeans sample database alias
□	neon/psAlias	psUser	JAAS Alias for Pet Store Sample

WebSphere Status

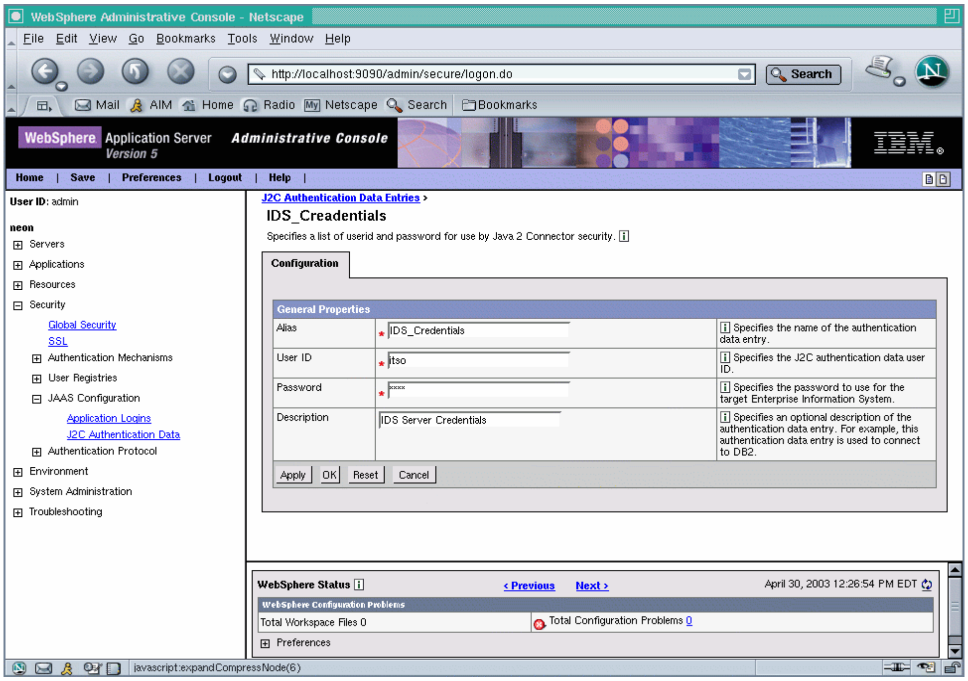
April 30, 2003 12:16:53 PM EDT

WebSphere Configuration Problems	
Total Workspace Files 0	Total Configuration Problems 0

⊞ Preferences

Figure 6-11 Adding a new J2C entry

3. Fill out the J2C fields and then click **OK**. See Figure 6-12. In our case we use:
   - **Alias:** IDS_Credentials
   - **User ID:** itso
   - **Password:** itso
   - **Description:** IDS Server Credentials


WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:9090/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
neon
Servers
Applications
Resources
Security
Global Security
SSL
Authentication Mechanisms
User Registries
JAAS Configuration
Application Logins
J2C Authentication Data
Authentication Protocol
Environment
System Administration
Troubleshooting
J2C Authentication Data Entries >
Configuration
General Properties
Apply OK Reset Cancel
WebSphere Status
< Previous Next >
April 30, 2003 12:26:54 PM EDT
WebSphere Configuration Problems
Total Workspace Files 0
Total Configuration Problems 0
Preferences
javascript:expandCompressNode(6)

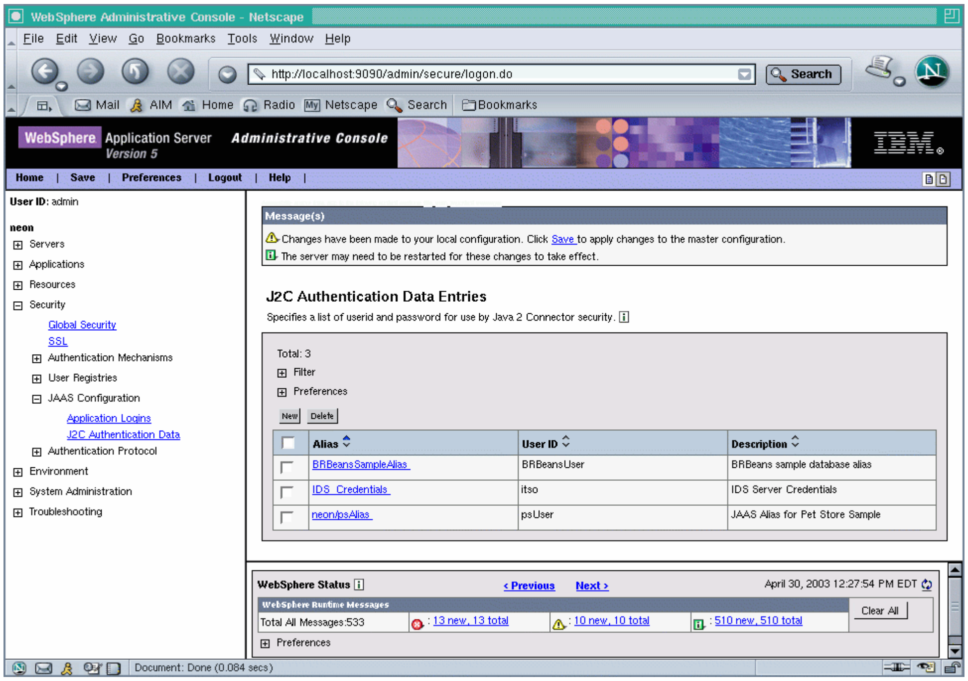
IDS_Creadentials

Specifies a list of userid and password for use by Java 2 Connector security.

Alias	* IDS_Credentials	Specifies the name of the authentication data entry.
User ID	* itso	Specifies the J2C authentication data user ID.
Password	* ****	Specifies the password to use for the target Enterprise Information System.
Description	IDS Server Credentials	Specifies an optional description of the authentication data entry. For example, this authentication data entry is used to connect to DB2.

Figure 6-12 Fill out the J2C entries

4. Now the new J2C entry is shown in the list. See Figure 6-13. Notice that a message box is displayed saying that we have to apply the changes to the master configuration. Click **Save**.



Changes have been made to your local configuration. Click Save to apply changes to the master configuration.

The server may need to be restarted for these changes to take effect.

J2C Authentication Data Entries

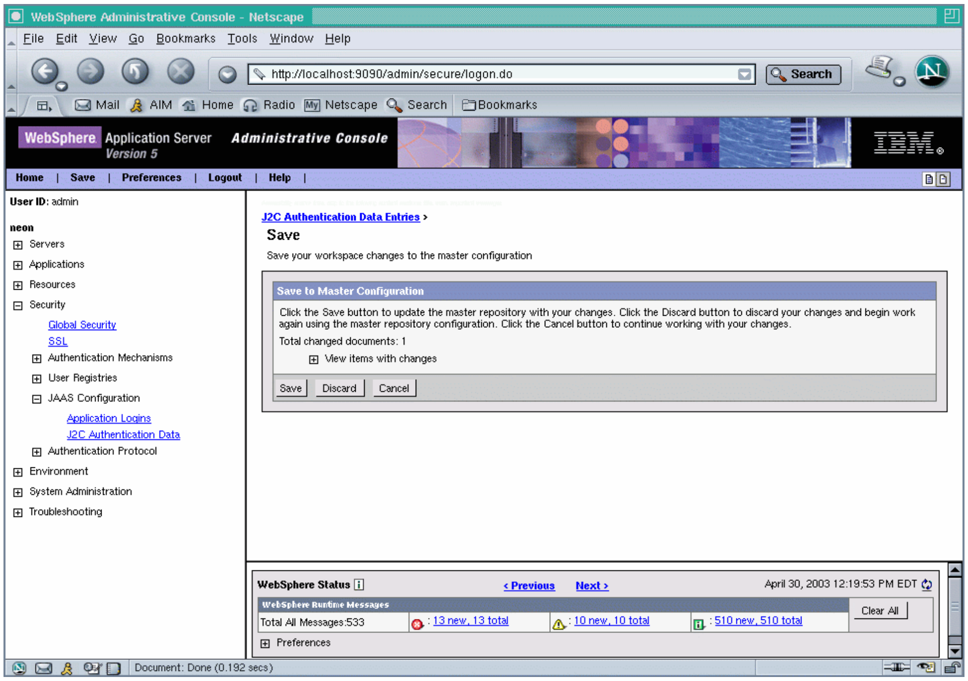
Specifies a list of userid and password for use by Java 2 Connector security.

Total: 3

□	Alias	User ID	Description
□	BRBeansSampleAlias	BRBeansUser	BRBeans sample database alias
□	IDS_Credentials	itso	IDS Server Credentials
□	neon/psAlias	psUser	JAAS Alias for Pet Store Sample

Figure 6-13 The added J2C entry is listed

5. Click the **Save** button to save the changes to the master repository. See Figure 6-14.


WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:9090/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help
neon
Servers
Applications
Resources
Security
Global Security
SSL
Authentication Mechanisms
User Registries
JAAS Configuration
Application Logins
J2C Authentication Data
Authentication Protocol
Environment
System Administration
Troubleshooting
Save
Discard
Cancel
Clear All
Preferences
Document: Done (0.192 secs)

User ID: admin

J2C Authentication Data Entries >

Save

Save your workspace changes to the master configuration

Save to Master Configuration

Click the Save button to update the master repository with your changes. Click the Discard button to discard your changes and begin work again using the master repository configuration. Click the Cancel button to continue working with your changes.

Total changed documents: 1

View items with changes

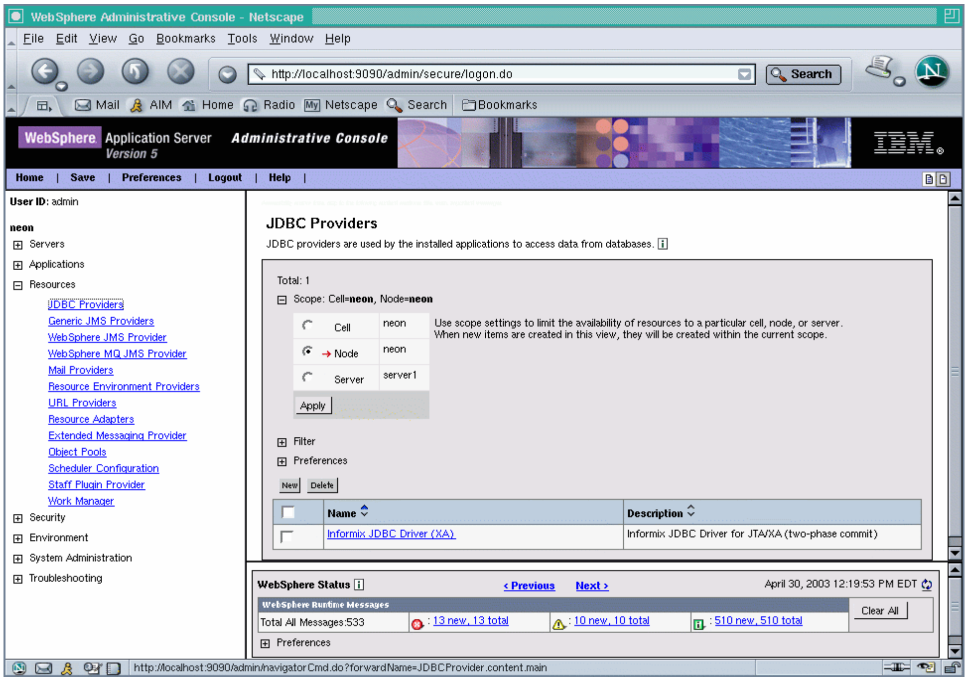
WebSphere Status < Previous Next > April 30, 2003 12:19:53 PM EDT

WebSphere Runtime Messages

Total All Messages:533 : 13 new, 13 total : 10 new, 10 total : 510 new, 510 total

Figure 6-14 Applying changes to master configuration

6. Now that we have an authentication method, we can create an Informix data source. To do that, click the **Resources** tab on the left frame and choose **JDBC Providers.** See Figure 6-15.


WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:9090/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Application Server Version 5 Administrative Console
Home | Save | Preferences | Logout | Help
neon
Servers
Applications
Resources
JDBC Providers
Generic JMS Providers
WebSphere JMS Provider
WebSphere MQ JMS Provider
Mail Providers
Resource Environment Providers
URL Providers
Resource Adapters
Extended Messaging Provider
Object Pools
Scheduler Configuration
Staff Plugin Provider
Work Manager
Security
Environment
System Administration
Troubleshooting
Apply
Filter
Preferences
New
Delete
Clear All
Preferences
http://localhost:9090/admin/navigatorCmd.do?forwardName=JDBCProvider.content.main

User ID: admin

JDBC Providers

JDBC providers are used by the installed applications to access data from databases.

Total: 1

Scope: Cell=**neon**, Node=**neon**

○ Cell	neon	Use scope settings to limit the availability of resources to a particular cell, node, or server. When new items are created in this view, they will be created within the current scope.
● Node	neon	
○ Server	server1	

□	Name	Description
□	Informix JDBC Driver (XA)	Informix JDBC Driver for JTA/XA (two-phase commit)

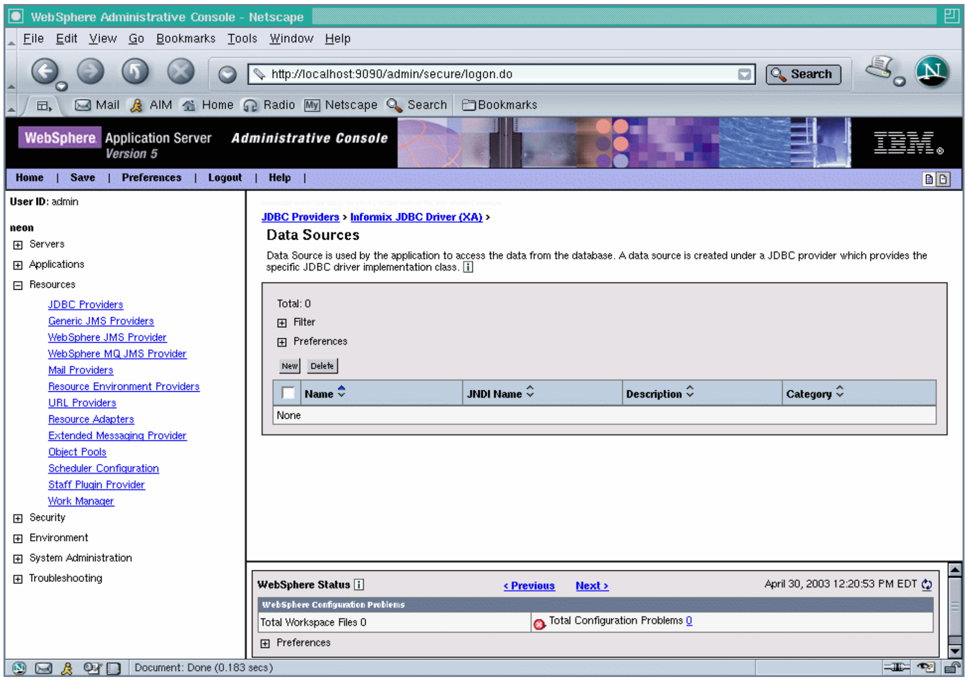
WebSphere Status < Previous Next > April 30, 2003 12:19:53 PM EDT

WebSphere Runtime Messages

Total All Messages:533 : 13 new, 13 total : 10 new, 10 total : 510 new, 510 total

Figure 6-15 Creating an Informix data source

7. Notice that the Informix JDBC provider is already listed (configured in Chapter 5, "Installing and configuring WebSphere Application Server" on page 103). Select this JDBC provider and its configuration is displayed. Scroll down and select **Data Sources** on the **Additional Properties** window. Then click the **New** button and the data source configuration window is displayed. See Figure 6-16.


WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:9090/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
neon
Servers
Applications
Resources
JDBC Providers
Generic JMS Providers
WebSphere JMS Provider
WebSphere MQ JMS Provider
Mail Providers
Resource Environment Providers
URL Providers
Resource Adapters
Extended Messaging Provider
Object Pools
Scheduler Configuration
Staff Plugin Provider
Work Manager
Security
Environment
System Administration
Troubleshooting
Document: Done (0.183 secs)

JDBC Providers > Informix JDBC Driver (XA) >

Data Sources

Data Source is used by the application to access the data from the database. A data source is created under a JDBC provider which provides the specific JDBC driver implementation class.

Total: 0

Filter

Preferences

New Delete

□	Name	JNDI Name	Description	Category
None				

WebSphere Status < Previous Next > April 30, 2003 12:20:53 PM EDT

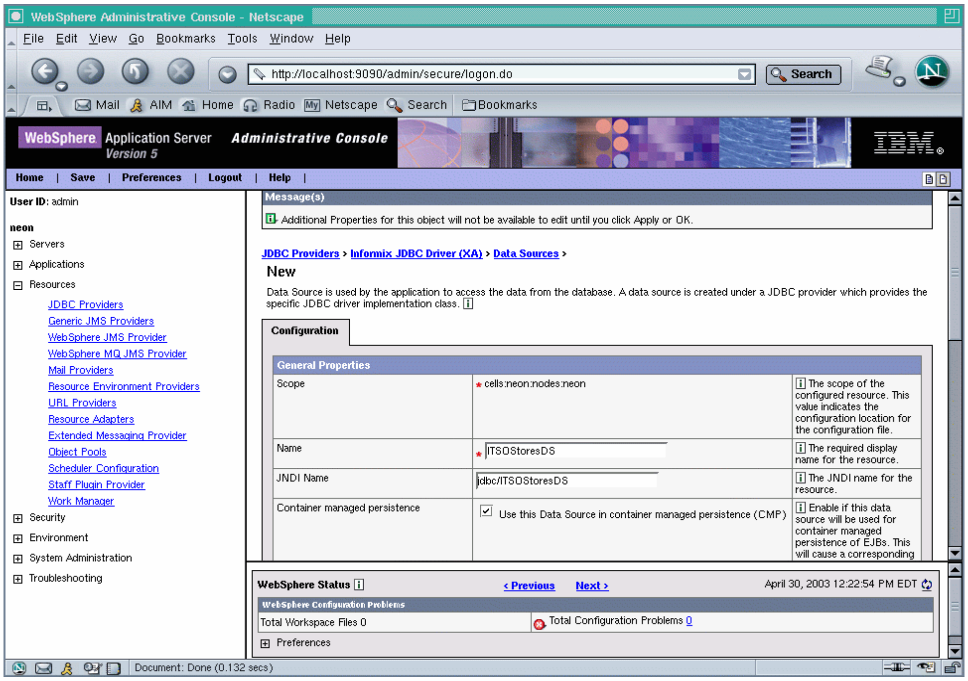
WebSphere Configuration Problems	
Total Workspace Files 0	Total Configuration Problems 0

Preferences

Figure 6-16 Data source configuration window

8. In this window, specify the following fields (showed with our settings):
   - **Name:** ITSOStoresDS
   - **JNDI Name:** jdbc/ITSOStoresDS
   - **Container managed persistence:** Select the check button.
   - **Component-managed Authentication Alias:** Select **IDS_Credentials** JAAS entry.
   - **Container-managed Authentication Alias:** Select **IDS_Credentials** JAAS entry.

   Click **OK** and save the changes to the master configuration (Message window). See Figure 6-17.


WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:9090/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Application Server Version 5
Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
neon
Servers
Applications
Resources
JDBC Providers
Generic JMS Providers
WebSphere JMS Provider
WebSphere MQ JMS Provider
Mail Providers
Resource Environment Providers
URL Providers
Resource Adapters
Extended Messaging Provider
Object Pools
Scheduler Configuration
Staff Plugin Provider
Work Manager
Security
Environment
System Administration
Troubleshooting
Document: Done (0.132 secs)

Message(s)

Additional Properties for this object will not be available to edit until you click Apply or OK.

JDBC Providers > Informix JDBC Driver (XA) > Data Sources >

New

Data Source is used by the application to access the data from the database. A data source is created under a JDBC provider which provides the specific JDBC driver implementation class.

Configuration

General Properties		
Scope	* cells:neon:nodes:neon	The scope of the configured resource. This value indicates the configuration location for the configuration file.
Name	* ITSOStoresDS	The required display name for the resource.
JNDI Name	jdbc/ITSOStoresDS	The JNDI name for the resource.
Container managed persistence	☑ Use this Data Source in container managed persistence (CMP)	Enable if this data source will be used for container managed persistence of EJBs. This will cause a corresponding

WebSphere Status < Previous Next > April 30, 2003 12:22:54 PM EDT

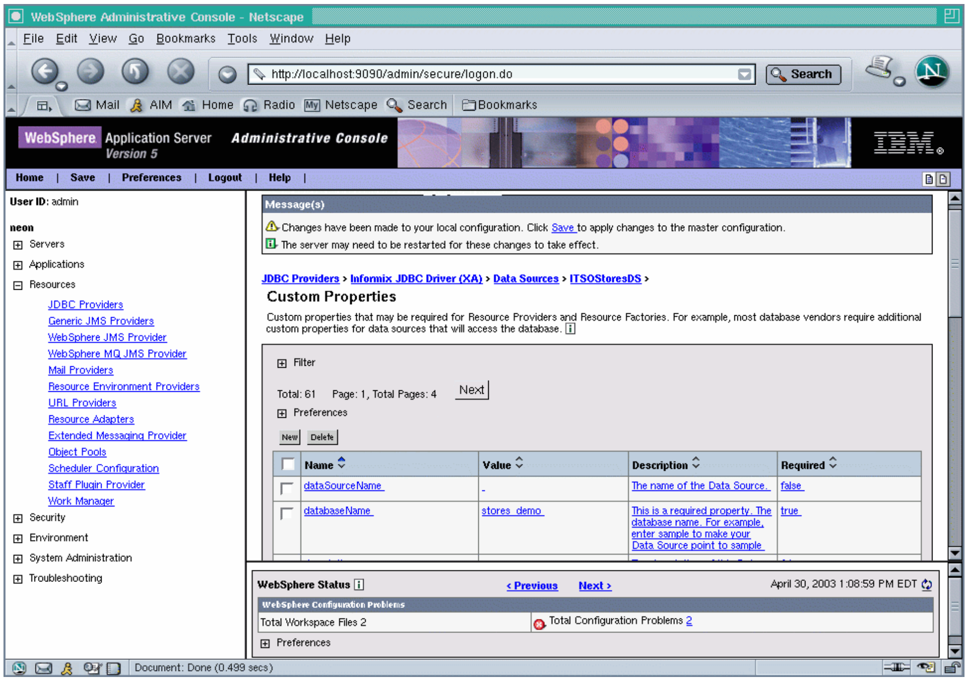
WebSphere Configuration Problems	
Total Workspace Files 0	Total Configuration Problems 0

Preferences

Figure 6-17 Message window

9. Select the data source created again (**JDBC Providers -> Informix JDBC Driver (XA) -> Data Sources -> ITSOStoresDS**). See Figure 6-18. Scroll down and select the **Custom Properties** in the **Additional Properties** window. Here we have to set the specific connection configuration for our Informix data source. The custom properties are composed of 4 configuration pages, but actually only 4 fields are mandatory, and these are the ones that we configured:

   – **databaseName:** stores_demo (page1)
   – **ifxIFXHOST:** 9.1.38.76 (page1)
   – **portNumber:** 1533 (page3)
   – **serverName:** demo_on (page4)



WebSphere Administrative Console - Netscape
File Edit View Go Bookmarks Tools Window Help
http://localhost:9090/admin/secure/logon.do
Search
Mail AIM Home Radio Netscape Search Bookmarks
WebSphere Application Server Version 5 Administrative Console
IBM
Home | Save | Preferences | Logout | Help |
User ID: admin
neon
Servers
Applications
Resources
JDBC Providers
Generic JMS Providers
WebSphere JMS Provider
WebSphere MQ JMS Provider
Mail Providers
Resource Environment Providers
URL Providers
Resource Adapters
Extended Messaging Provider
Object Pools
Scheduler Configuration
Staff Plugin Provider
Work Manager
Security
Environment
System Administration
Troubleshooting
Document: Done (0.499 secs)

Message(s)

Changes have been made to your local configuration. Click Save to apply changes to the master configuration.

The server may need to be restarted for these changes to take effect.

JDBC Providers > Informix JDBC Driver (XA) > Data Sources > ITSOStoresDS >

Custom Properties

Custom properties that may be required for Resource Providers and Resource Factories. For example, most database vendors require additional custom properties for data sources that will access the database.

⊞ Filter

Total: 61 Page: 1, Total Pages: 4 Next

⊞ Preferences

New Delete

□	Name	Value	Description	Required
□	dataSourceName		The name of the Data Source.	false
□	databaseName	stores_demo	This is a required property. The database name. For example, enter sample to make your Data Source point to sample	true

WebSphere Status < Previous Next > April 30, 2003 1:08:59 PM EDT

WebSphere Configuration Problems	
Total Workspace Files 2	Total Configuration Problems 2

⊞ Preferences

Figure 6-18 Selecting the data source

10.Go back to the first configuration windows of the **ITSOStoresDS** data source and click **OK**. Save the changes to the master configuration and the process is complete.

We now have the connections defined and have configured the Informix Data Source on WebSphere Application. We are now ready to develop a sample application and demonstrate the integration of WebSphere and Informix Dynamic Server.

7

Working with IDS and WebSphere

In this chapter we present a number of advanced topics regarding application development and the use of WebSphere and IDS. We demonstrate the robust and powerful capabilities of WebSphere and its integration with IDS. These capabilities will be important to you as you design, architect, and implement your application development environment. Built on open standards, an environment based on IDS and WebSphere can provide you with the power and flexibility you need to be a fast moving and highly competitive business.

We describe a number of advanced topics, but we also maintain a practical approach by guiding you through the development of sample applications using the technologies and capabilities we describe. The following topics are discussed:

- Creating an application using Database Web Pages
- Creating a Java Server Page
- Creating a J2EE container managed persistent bean
- Working with Java Message Services and IDS
- Creating a sample application to demonstrate IDS and WebSphere capabilities
- Managing transactions

7.1 Introduction to the sample applications

WebSphere Studio Application Developer (WSAD) is the product used to develop the sample applications in this chapter. These sample application use the above described technologies and functional capabilities. They demonstrate capabilities such as performing select, insert, update, and delete operations on IDS. They also demonstrate development using Database Web Pages, Java Beans, Container Managed Persistence, and Java Messaging Services for running in a physically or logically distributed environment.

There are four sample applications, for which you are given step-by-step instructions to develop, in this redbook. In addition, for those who may prefer not to go through the exercises, we have provided already developed sample applications that are downloadable from the Redbooks Web site. See Appendix D, "Additional material" on page 343 for instructions on how to download those applications.

Here are the names and a brief description of the four sample applications. All of them access the Stores_demo database on IDS:

1. **SelectApplication:** This is a sample application named "ITSOStoresDBPages" that only performs a ***select*** operation against IDS. It was developed with Database Web Pages. The development and deployment of this application is described in Chapter 4, "Installing and configuring WebSphere Studio V5" on page 85.
2. **FullApplication:** This is a sample application that is also called "ITSOStoresDBPages". (Be careful — it has the same name as the SelectApplication!) It is actually the same application base, but has been expanded and now performs a ***select, insert, update, and delete*** operation against IDS. It was also built with Database Web Pages.
3. **ITSOStores:** This is a sample application called "ITSOStores". It is similar to the FullApplication described above, but is developed with different technology and has expanded capabilities. For example, it was not developed with Database Web Pages, but rather includes the use of such things as Java Beans and Container Managed Persistence. It was developed to run in a single server environment.
4. **ITSOStoresJMS:** This is a sample application called "ITSOStoresJMS". It is similar to the sample application ITSOStores described above, but has been further enhanced to run in a distributed environment (although it will also run in a single server environment). To enable it to run in a distributed environment we have added Java Messaging Services.

Going through the stet-by-step instructions and developing these sample applications with WebSphere Application Developer, and deploying them on WebSphere Application Server, will be great education and a great training exercise for you. You will then have first-hand knowledge of these products and their capabilities.

7.2 Extending the Database Web Pages sample application

In Chapter 4, "Installing and configuring WebSphere Studio V5" on page 85, we described how to create a Database Web Page application to perform a *select* operation on the Stores_demo database in IDS. That application was intended to demonstrate the connectivity between WSAD and IDS, and get you started towards a better understanding of WSAD, WAS, and IDS.

In this section we develop a more complex application, using some of the advanced capabilities available to you with IDS and WebSphere. We named it the **FullApplication**. For example, it will demonstrate select, insert, update, and delete operations against IDS. However, it still uses Database Web Pages. We will extend the sample application developed in Chapter 4, “Installing and configuring WebSphere Studio V5” on page 85 to demonstrate this. Later in this chapter we develop sample applications based on different technology.

To extend the sample application using Database Web Pages, we will use WSAD to get you more familiar with its capabilities for application development. WSAD generates application code and masks many of the technical development details, and thereby reduces the requirement for skilled Java programmers. This should, in turn, minimize the application development time. This application is developed and documented with the intent that it be understandable by non-Java programmers, and required minimal use of skilled Java programming.

We also demonstrate advanced capabilities in this chapter, such as distributed applications, messaging, transaction management, and application persistence. Although in a simple application in this redbook, these capabilities enable the implementation of a robust and flexible application development environment.

7.2.1 Steps to extend the sample application

In this application there are four modules to demonstrate our four database operations of search, insert, update, and delete. We show the steps to extend the initial SelectApplication. First we will create each module and then integrate them using one Java Server Page (JSP), which we call *MainMenu.jsp.*

Search Module:

We have already described the steps for creating a query application in Chapter 4, and it is the same as our search module here. So, we will use those same steps again in this section.

1. Follow step “Open Web Sphere Studio Application Developer.” on page 89 to step Step 30 on page 97.

Database Model:

To connect to the database from WSAD, we must provide information such as the name of the database, user ID, password, and host, as shown in Figure 4-7 on page 93. Typically, for every connection, we need to enter all information again and again. But instead of that, we will describe how to create a Database Model and then connect to the database just by referring to the database model. In WSAD, do the following:

2. Go to **Data** perspective (**Windows -> Open Perspective -> Other -> Data**).
3. Go to **DB Servers** view (**Windows -> Show View -> Other -> DB Servers**).
4. In **DB Servers** view we are going to create a connection. Right-click in **DB Servers** view and select **New Connection**. Wizard **New** will be opened to enter the connection details.
5. Enter **Connection name** as *ConnToIDS*, **Database** as *stores_demo,* **User ID** as *itso*, **Password** as password of the itso user. Select **Database vendor type** as *Informix Dynamic Server, V 9.3*, **JDBC driver** as *INFORMIX JDBC NET DRIVER*, **Host** as ipaddress of the IDS server, **Port number** as port number of the server instance demo_on, **Server name** as *demo_on*, **Class location** as the *<JDBC Driver Installed Directory>\lib\ifxjdbc.jar.* After entering all the values, please cross check values with the Figure 4-7. Click the **Finish** button. Refer to step Step 13 on page 92 and read the note below it. A new connection **ConnToIDS** will be created. You can find it in the **DB Servers** view. We have now completed the creation of the connection.

6. We are going to import the schema of stores_demo into WSAD workspace using the connection **ConnToIDS**. Right-click the connection **ConnToIDS** and then click **Import to Folder...**. Wizard **Import** will be opened.

7. Enter **Folder** as *Databases.* Click the **Finish** button. As folder *Databases* does not exist, the confirmation dialogue box to create a folder will be opened. Click the **Yes** button. It will create folder called Databases.

8. Go to the **Data Definition** view. Find the **stores_demo**, database model in the Databases folder. If you expand it, you can find the schema of the stores_demo database. Now we are ready with a database model, that is connected to the stores_demo database in IDS.

Insert Module:

9. Go to the Web perspective.(**Window -> Open Perspective -> Web**).

10. Expand project **ITSOStoresDBPagesWeb**. Find the WebContent folder under ITSOStoresDBPages.

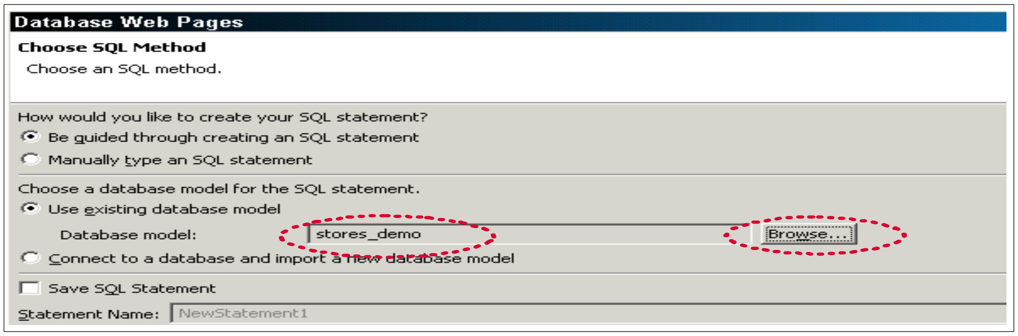
11. Right-click the **WebConnect** folder and then click **New -> Other**. A Wizard **New** will be opened.

12. In the Wizard select **Web** from left pane and then **Database Web Pages** from the right pane. Click the **Next** button. The **Database Web Pages** wizard will be opened.

13. Enter name of the **Java package** as *itso.stores*. Select **SQL Statement Type** as *Insert Statement*, **Model** as *IBM Database Access Tag Library - Insert Statement*. Click the **Next** button. The contents of **Data Base Web Pages** Wizard will be changed to **Choose SQL Method**.

14. Select **How would you like to create your SQL statement?** as *Be guided through creating an SQL statement* and **Choose a database model for the SQL statement** as *Use existing database model.* Click the **Browse** button, to select existing database model **stores_demo** under Databases folder. The Wizard **Data resource selection**, will then be opened.

15. Select **Databases -> stores_demo**. Click the **OK** button. Now you will be back to the **Data Base Web Pages** Wizard, as shown in Figure 7-1.



Database Web Pages

Choose SQL Method

Choose an SQL method.

How would you like to create your SQL statement?

● Be guided through creating an SQL statement

○ Manually type an SQL statement

Choose a database model for the SQL statement.

● Use existing database model

Database model: stores_demo Browse...

○ Connect to a database and import a new database model

□ Save SQL Statement

Statement Name: NewStatement1

Figure 7-1 Choosing database model

16. Click the **Next** button. The contents of **Data Base Web Pages** Wizard will be changed to **Construct An SQL Statement**.

17. Find two tabs **Tables** and **Insert**, in this page. Select **itso -> itso.customer** table in the left pane of the **Tables** tab. Click the button **>** to select **itso.customer** table.

18. Go to the **Insert** tab. You will find the list of column names of the **itso.customer** table. Under the **Insert** tab, we need to map each column name of the **customer** table with a host variable name. Host variables holds customer information.

19. Enter **customer_num**’s value as 0. All the cells under **Value**, will become editable by double-clicking them.

> **Note:** Field **customer_num** is a primary key of customer table. So unique numbers have to be inserted. As this is of type *serial*, if you insert a record with **customer_num** as zero, IDS will generate a unique customer number and inserts a new record into IDS.

20. Map column **fname** with a host variable by building an expression. Double-click the value cell of **fname** and a combo box will be displayed. Select **Build Expression...** from the combo box list. The **Expression Builder** Wizard will be displayed.

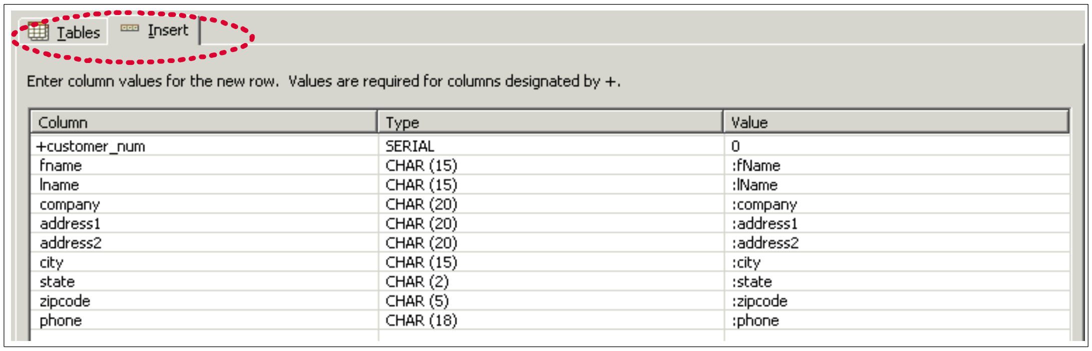
21. In the **Expression Builder** Wizard, select the type of expression to build as **Constant - numeric, string or host variable** and click **Next** as shown in Figure 4-10 on page 95. The contents of the **Expression Builder** Wizard will be changed to **Constant Options Page**.

22. Select Constant type as **String constant** and click **Next**. The contents of the **Expression Builder** Wizard will be changed to **String Constant Builder Page** to take the name of the variable which contains customer number entered by the user.

23. Enter Host variable name as *fName* and select **String constant type** as **Host variable name**. Click the **Finish** button. You will be taken back to the **Database Web Pages** Wizard.

24. Repeat the Step 20 on page 147 to Step 23 on page 147, for the remaining column names, **lname**, **company**, **address1**, **address2**, **city**, **state**, **zipcode** and **phone**. But take care during Step 23 on page 147 to enter unique Host variable name.

25. After finishing the above steps, the contents under the **Insert** tab should look as shown in Figure 7-2. Click the **Next** button in **Database Web Pages** wizard. Contents of **Database Web Pages** wizard will be changed to **SQL Statement Page**.



Enter column values for the new row. Values are required for columns designated by +.

Column	Type	Value
+customer_num	SERIAL	0
fname	CHAR (15)	:fName
lname	CHAR (15)	:lName
company	CHAR (20)	:company
address1	CHAR (20)	:address1
address2	CHAR (20)	:address2
city	CHAR (15)	:city
state	CHAR (2)	:state
zipcode	CHAR (5)	:zipcode
phone	CHAR (18)	:phone

Figure 7-2 Mapping columns names - Insert Module

26. Click **Next** button in **SQL Statement Page (Database Web Page)**. Contents of **Database Web Pages** wizard will be changed to **Runtime Connection Pages**.

27. Select database connection as **Use driver manager connection**.
Enter **Driver name** as *com.informix.jdbc.IfxDriver,* **URL** as *jdbc:informix-sqli://thorium:<portnumber>/stores_demo:INFORMIXSERVER=demo_on;* , **User ID** as *itso*, **Password** as *<password of user itso>,* **Re-enter password** as *<password of user itso>.* Click the **Next** button. Contents of **Database Web Pages** wizard will be changed to **Controller Page**.

28. Select radio button **Do not use a Front Controller**. Click the **Next** button. Contents of **Database Web Pages** wizard will be changed to **Design the Input Form.** This form is used to specify our customer number of interest.

29. Click the **Next** button. Contents of **Database Web Pages** wizard will be changed to **Design the Insert View**. This form is used to display the result of insert operation.

30. Click the **Next** button. Contents of **Database Web Pages** wizard will be changed to **Specify Prefix**.

31. Click the **Finish** button.

32. It will generate the customerInputForm.html and customerInsertView.jsp under Web Content folder in the project **ITSOStoresDBPagesWeb**. Development of insert module is finished.

Update Module:

33. Development of this module is very similar to the development of Insert module. So we will discuss only the steps which differ.

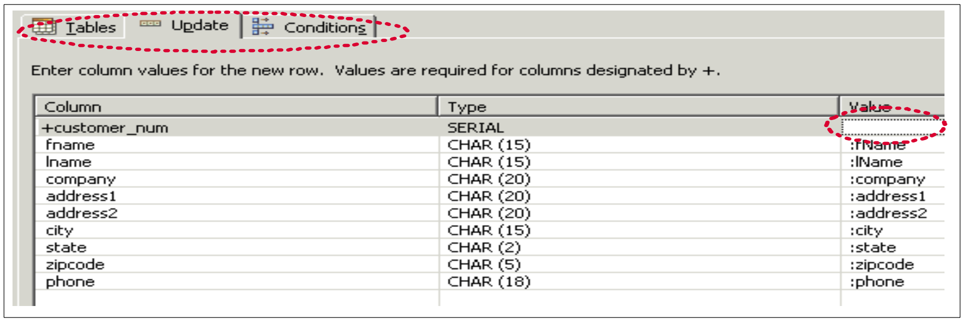
34. Perform Step 9 on page 146 to Step 16 on page 146, but during Step 13 on page 146, select **SQL Statement Type** as *Update Statement*, **Model** as *IBM Database Access Tag Library - Update Statement*.

35. Find three tabs **Tables, Update** and **Conditions** in **Construct An SQL Statement (Database Web Pages)** page. Select **itso -> itso.customer** table in the left pane of the **Tables** tab. Click the button **>** to select **itso.customer** table.

36. Go to the **Update** tab. You will find the list of column names of **itso.customer** table. We need to map each column name of **customer** table with a host variable name. Host variables holds customer information.

37. We have shown the mapping of fName with a Host variable, from Step 20 on page 147 to Step 23 on page 147. Similarly follow the same steps to map **lname**, **company**, **address1**, **address2**, **city**, **state**, **zipcode** and **phone**. But don't enter anything against **customer_num**. Because it is primary key.

38. After finishing the above steps, contents under **Update** tab should look as in Figure 7-3.



Enter column values for the new row. Values are required for columns designated by +.

Column	Type	Value
+customer_num	SERIAL	
fname	CHAR (15)	:fName
lname	CHAR (15)	:lName
company	CHAR (20)	:company
address1	CHAR (20)	:address1
address2	CHAR (20)	:address2
city	CHAR (15)	:city
state	CHAR (2)	:state
zipcode	CHAR (5)	:zipcode
phone	CHAR (18)	:phone

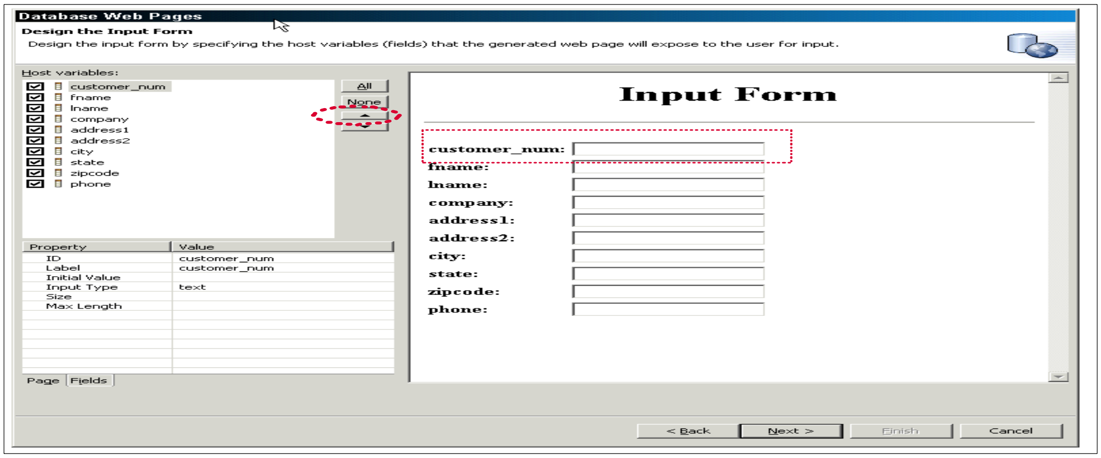
Figure 7-3 Mapping columns names - Update Module

39. Go to **Conditions** tab. Specify the condition to update the record whose customer_num is same as the number entered by user. To specify this condition, refer Step 16 on page 93 to Step 20 on page 94.

40. Now we specified condition, so we are ready to finish the Update module.

41. Click the **Next** button. Contents of **Database Web Pages** wizard will be changed to **SQL Statement Page**.

42. The remaining steps are very similar to Insert module. Follow the Step 26 on page 147 to Step 31 on page 148. but during Step 28 on page 147, design the input form as shown in Figure 7-4. The **customer_num** should be the first field on the page. Use the up arrow button as highlighted, to adjust the placement of the field **customer_num**.



Design the Input Form

Design the input form by specifying the host variables (fields) that the generated web page will expose to the user for input.

Host variables:

- ☑ customer_num
- ☑ fname
- ☑ lname
- ☑ company
- ☑ address1
- ☑ address2
- ☑ city
- ☑ state
- ☑ zipcode
- ☑ phone

Property	Value
ID	customer_num
Label	customer_num
Initial Value	
Input Type	text
Size	
Max Length	

Input Form

customer_num:

fname:

lname:

company:

address1:

address2:

city:

state:

zipcode:

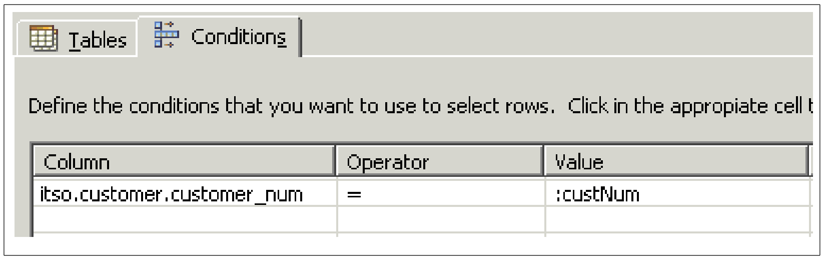
phone:

Figure 7-4 Input form - Update modules

43. As part of update module, one input form and update view files will be created. Update module also is finished.

Delete Module:

44. Development of the Delete Module is very similar to previous modules. We will discuss only those items specific to the delete module. Select **SQL Statement Type** as *Delete Statement*, **Model** as *IBM Database Access Tag Library - Delete Statement* and specify the same condition as Update module, during construction of SQL Statement as shown in Figure 7-5.



Define the conditions that you want to use to select rows. Click in the appropiate cell t

Column	Operator	Value
itso.customer.customer_num	=	:custNum

Figure 7-5 Delete condition

45. As part of the delete module, two files will be created. One to take inputs from user and the other to perform the delete operation and to display results to the user. With this we have finished the development of different modules. Now we must integrate these modules.

7.2.2 Creating a JSP to integrate the application modules

Creation and design of the JSP is Wizard driven, and straight forward. We are going to create MainMenu.jsp which contains links to the search, insert, update and delete modules.

Steps to integrate all the modules:

46. Go to Web Perspective.

47. Click **File -> New -> JSP file**.

48. Select **Folder** as *ITSOStoresDBPagesWeb -> Web Content,* **File Name** as *MainMenu.*

49. Click the **Finish** button.

50. Go to **J2EE Navigator** view. Find MainMenu.jsp under the folder ITSOStoresDBPagesWeb -> Web Content.

51. Close all files opened in WSAD editor.

52. Open only MainMenu.jsp.

53. Click the **Design** tab of MainMenu.jsp.

> **Note: Design mode** of JSP helps us to design the page. If you have any problem in reading the content you have typed in design mode of MainMenu.jsp, then download the Master.css file from the Redbooks additional material Web site. Import it into your workspace under the folder, **ITSOStoresDBPagesWeb -> Web Content** and add the following code to the source of MainMenu.jsp in between <head> </head>:
>
> ```
> <STYLE type="text/css">
> @IMPORT url("/ITSOStoresDBPagesWeb/Master.css");</STYLE>
> ```

54. Replace the text **Place Test.jsp's content here** with **Welcome to ITSOStores**. This is going to be the title of our page. Remember, to do these tasks we must be in design mode of MainMenu.jsp. If you are not able to find the text **Place Test.jsp's content here** on MainMenu.jsp, refer to the foregoing note.

55. From the **J2EE Navigator** view, drag and drop the input file of the search module onto WSAD onto the design page of MainMenu.jsp. The **Insert File** dialogue box will pop up, providing different options for inserting a file as shown in Figure 7-6.

56. Select **Insert a link to this file** and click the **OK** button.

57. Find the inserted link to the search module. The name of the link will be **Input Form**. Edit it to **Search**. Links and titles can be aligned by using tables. But we will just a create working model.

58. Repeat Step 55 on page 150 to Step 57 on page 150, for rest of the modules **Insert**, **Update**, **Delete** and name the links as **Insert**, **Update**, **Delete**. After doing the formatting, the main page should look like Figure 7-7. Save and close the MainMenu.jsp.

59. To provide navigation from modules to Main page, close the MainMenu.jsp and open each input form and view form, drag and drop the MainMenu.jsp on to these files to provide navigation from these pages to MainMenu.jsp.

Congratulations, the application is now ready to deploy on either the WSAD or WS Application Server.

> **Note:** This section takes you through step-by-step instructions to develop a sample application using Database Web Pages. If you prefer not to go through the exercise of developing your own sample application, you can simply download the already developed demo sample application file called *SampleApps.Zip* from Appendix D, “Additional material” on page 343.
>
> Inside the zipped download file are six folders. Open the folder named *FullApplication*. Inside is the actual sample demo application, called *ITSOStoresDBPages.ear.* First, import this file into your WSAD workspace. Then follow the development instructions in Steps 39 to 44 in the Section “A sample application using Database Web Pages” on page 88.

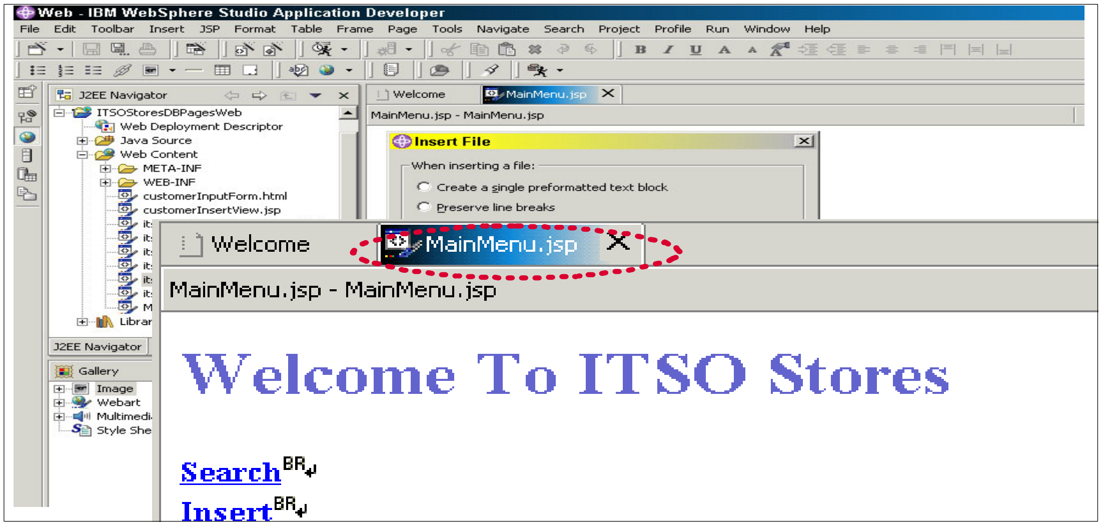
7.2.3 Deploying the application

In this section we explain how you can deploy the application.

Deploying the application in WSAD

1. To deploy the application on WebSphere Studio, follow the instructions in “Deploying the Web application in the test server:” on page 99.
2. Restart the server and access the application using the following URL:

```
http://<I P Address of test server>:9080/ITSOStoresDBPagesWeb/MainMenu.jsp
```


Web - IBM WebSphere Studio Application Developer
File Edit Toolbar Insert JSP Format Table Frame Page Tools Navigate Search Project Profile Run Window Help
J2EE Navigator
ITSOStoresDBPagesWeb
Web Deployment Descriptor
Java Source
Web Content
META-INF
WEB-INF
customerInputForm.html
customerInsertView.jsp
J2EE Navigator
Gallery
Image
Webart
Multimedi
Style She
Welcome
MainMenu.jsp
MainMenu.jsp - MainMenu.jsp
Insert File

When inserting a file:

○ Create a single preformatted text block

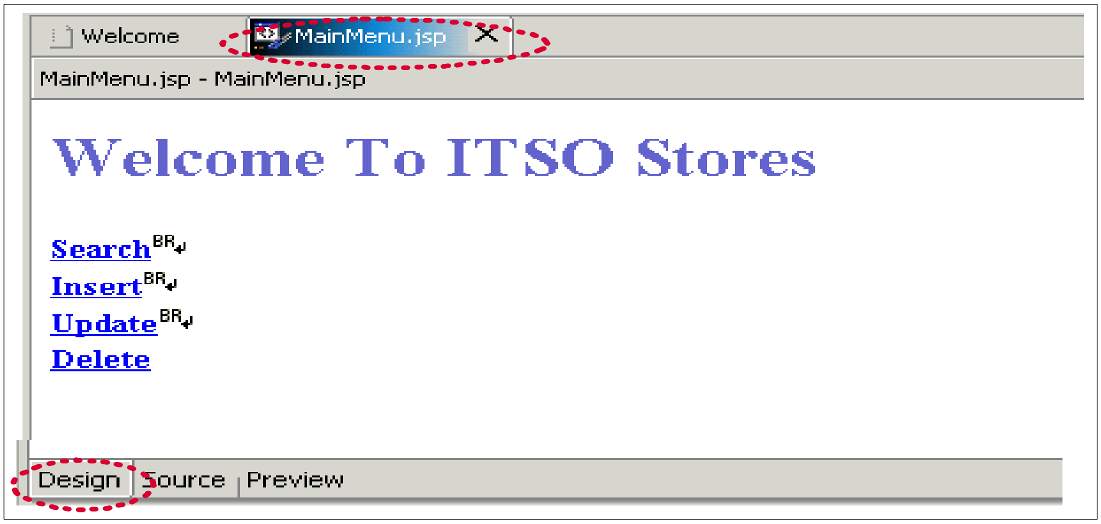
○ Preserve line breaks

Welcome To ITSO Stores

Search BR

Insert BR

Figure 7-6 Inserting a link in Main Menu



Welcome To ITSO Stores

Search BR

Insert BR

Update BR

Delete

Figure 7-7 Design of Main Page

Congratulations, you have deployed the sample application on WSAD application server.

Deploying an application on the Application Server

1. Get ITSOStoresDBPages.ear file from WS Studio and follow the instructions in "Deploy the ITSOStoresDBPages application in WAS" on page 115. Access the application using the following URL:

```
http://<I P Address of application server>:9080/ITSOStoresDBPagesWeb/MainMenu.jsp
```

7.3 Sample application: Container Managed Persistent Bean

In this section we describe a sample application using a Container Managed Persistent Bean.

7.3.1 The ITSOStores sample application

This sample application (ITSOStores) is designed to demonstrate the connectivity of IDS using a J2EE container managed persistent bean from WebSphere. A Container Managed Persistent (CMP) Bean is mapped with the customer table in IDS. This application performs search, insert, update, and delete operations on the customer table in the Stores_demo database.

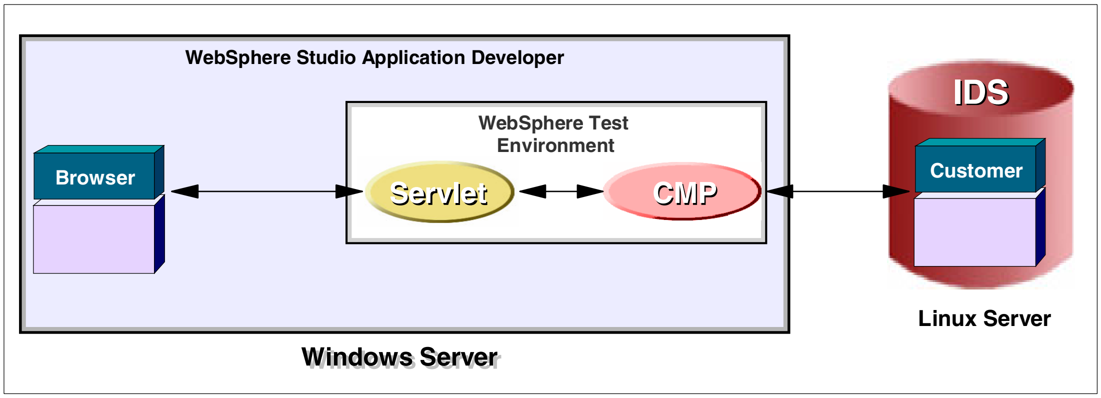

WebSphere Studio Application Developer
WebSphere Test Environment
Browser
Servlet
CMP
IDS
Customer
Linux Server
Windows Server

Figure 7-8 Architecture of ITSOStores applications

Components in the ITSOStores application

Around the CMP bean *customer*, some Java Server Pages (JSP) and Servlets are added to perform search, insert, update, and delete operations. For example, to perform a search operation, a JSP is added to accept search criteria such as the customer number. These details are passed to a servlet called Search.java, which uses the CMP to perform the search operation. The results are passed to another JSP, SearchResults.jsp, using ResultBean.java. In a similar fashion, the remaining files are added to build a complete application. Utility.java is used to perform common/frequently used operations across applications.

7.3.2 Steps for creating a CMP bean

In this section we provide the steps to create the `customer` entity bean.

1. Go to J2EE perspective.
2. Click **File -> New -> Enterprise Application Project**.
3. Select **J2EE1.3 Enterprise Application project** and click **Next**.
4. Enter **Enterprise application project name** as *ITSOStores.* Uncheck **Application client module**. Click the **Finish** button.

5. Right-click the project **ITSOStoresEJB** under **EJB Modules**.
6. Select **New -> Enterprise Bean**. **Enterprise Bean Creation** and a Wizard will pop up.
7. Select **EJB Project** as *ITSOStoresEJB.* Click **Next**.
8. Select **Enterprise Bean with container-managed persistence (CMP) fields** and *CMP 2.0 Bean.* Enter bean name as *customer.* Default package as *itso.stores.* Click **Next**.
9. Select *Remote client view* and *Local client view*. Click the button **Add,** to CMP attributes. **Create CMP Attribute** and a Wizard will pop up.
10. Enter **Name** as *customer_num* and **Type** as *int.* Select **Key field** to specify this field as a primary key. Click **Apply**.
11. Repeat Step "Expand project ITSOStoresDBPagesWeb. Find the WebContent folder under ITSOStoresDBPages." on page 146, for entering fName, lName, company, address1, address2, city, state, zipcode, and phone. All these are CMP fields of type *String.* For these fields uncheck **Key field**. After entering all CMP fields, click the **Close** button on **Create CMP Attribute** Wizard.
12. Click the **Finish** button, and the CMP bean is created.

Mapping the CMP Bean customer to the database table customer

13. Right-click the project ITSOStoresEJB and select **Generate -> EJB to RDB Mapping**.
14. Select **Create a new backend folder** and click **Next**.
15. Select **Meet In The Middle** and click **Next**.
16. Enter the database connection parameters the same as described in Step "Enter Connection name as ConnToIDS, Database as stores_demo, User ID as itso, Password as password of the itso user. Select Database vendor type as Informix Dynamic Server, V 9.3, JDBC driver as INFORMIX JDBC NET DRIVER, Host as ipaddress of the IDS server, Port number as port number of the server instance demo_on, Server name as demo_on, Class location as the <JDBC Driver Installed Directory>\lib\ifxjdbc.jar. After entering all the values, please cross check values with the Figure 4-7. Click the Finish button. Refer to step Step 13 on page 92 and read the note below it. A new connection ConnToIDS will be created. You can find it in the DB Servers view. We have now completed the creation of the connection." on page 145, or as described in previous sections, and click **Next**.
17. Select **itso -> customer** from the list of tables and click **Next**.
18. Select **None**.
19. Map.mapxmi file will be opened in editor. In the left pane, Enterprise Beans details will be there, and in the right pane, relational database table details will be there. EJB project **ITSOStoresEJB** and relational database stores_demo are already mapped. Drag CMP **customer** onto the relational database table **customer**. Now CMP and table are mapped. In the same way, map attributes of CMP and columns of the database table.
20. Save the file Map.mapxmi and close it.

Generating Deploy and RMIC Code

21. Right-click EJB Project **ITSOStoresEJB** and select **Generate -> Deploy and RMIC Code...** A Wizard to **Generate Deploy and RMIC Code** will pop up.
22. Select **customer** bean and click **Finish**.

Completing the deployment descriptor

Before we can test the entity bean, we must update the deployment descriptor with WebSphere-specific binding information:

23. In the J2EE Hierarchy view, double-click the `ItsoStoresEJB` module (or select *Open With -> Deployment Descriptor Editor* (context).

24. In the Overview pane of the deployment descriptor editor, we add two values for the CMP Factory Connection Binding:

    – For the JNDI name, enter `jdbc/ITSOStoresDS`. This is the JNDI name we used for the data source definition during the setup of the WebSphere Test Environment

    – For the Container authorization type, select *Per_Connection_Factory*.

25. Check that the correct current Backend ID is selected (`INFORMIX_V93_1`). The Container Managed Persistent bean is ready to test and use in the application.

Congratulations, you have succeeded in creating a CMP entity bean.

> **Note:** There is a potential problem inserting a new record into customer table using CMP:
>
> In the table customer, customer_num is of type *serial* and it is a primary key. IDS should generate a unique value for customer_num if a record is inserted with customer_num as 0. However, when we tried to insert a record with customer_num as zero, it did not function properly. So we generated a unique key before calling CMP, by using a sequence database object and then inserted a new record. For more details, refer to 13.1.5, “Using sequence objects rather than serial data type” on page 307.

Download the application

Download the demo sample application file called *SampleApps.Zip* from Appendix D, “Additional material” on page 343. Inside the zipped download file are four folders. Open the folder named *ITSOStores*. Inside is the actual sample demo application, called *ITSOStores.ear.*

7.3.3 Deploying the application

In this section we explain how you can deploy the application.

Deploying the application in WSAD

Prerequisites:

1. Configure Informix data source **ITSOStoresDS**, under Informix JDBC Provider. For details, refer to “Detailed steps to configure an Informix Data Source on WSAD” on page 132.

Steps to deploy application in WSAD:

2. Import the *ITSOStores.ear* file into workspace. You should see projects named ITSOStoresWeb and ITSOStoresEJB in your workspace.
3. Right-click ITSOStoresWeb project and click **Properties**. Properties of ITSOStoresWeb Wizard will pop up.
4. Select **Java Build Path** from the left pane.
5. In the right pane, go to the **Projects** tab and select ITSOStoresEJB.
6. Go to the **Libraries** tab and click the **Add External Jars** button to select ifxjdbc.jar and ifxjdbcx.jar.
7. Add application ITSOStores, to test server.
8. Restart the test server and the new application should be functional.

Congratulations, you have developed a sample application using a CMP Bean!

Deploying the application in WAS

Prerequisites to deploy in WAS:

9. Configure Informix JDBC Provider in the application server. For details, refer to 5.2.1, “Configuring the Informix JDBC Provider” on page 114.
10. Configure data source ITSOStoresDS, under Informix JDBC Provider. For details, refer to 6.4.3, “Configure Informix Data Source on WebSphere Application Server” on page 136.

Steps to deploy the application in WAS:

Deploy the application in *ITSOStores.ear.* Refer to instructions in “Deploy the ITSOStoresDBPages application in WAS” on page 115. Start the application with the URL:

```
http://<ipaddress of application server>:9080/ITSOStoresWeb/MainMenu.jsp
```

The sample application using a CMP Bean

Figure 7-9 depicts the Main Menu page for the ITSO Stores Demo sample application. It looks more sophisticated than the previous sample applications. We have used some of the capabilities of WebSphere Studio Application Developer to create an interface for the user that is more graphical and easier to use and understand.

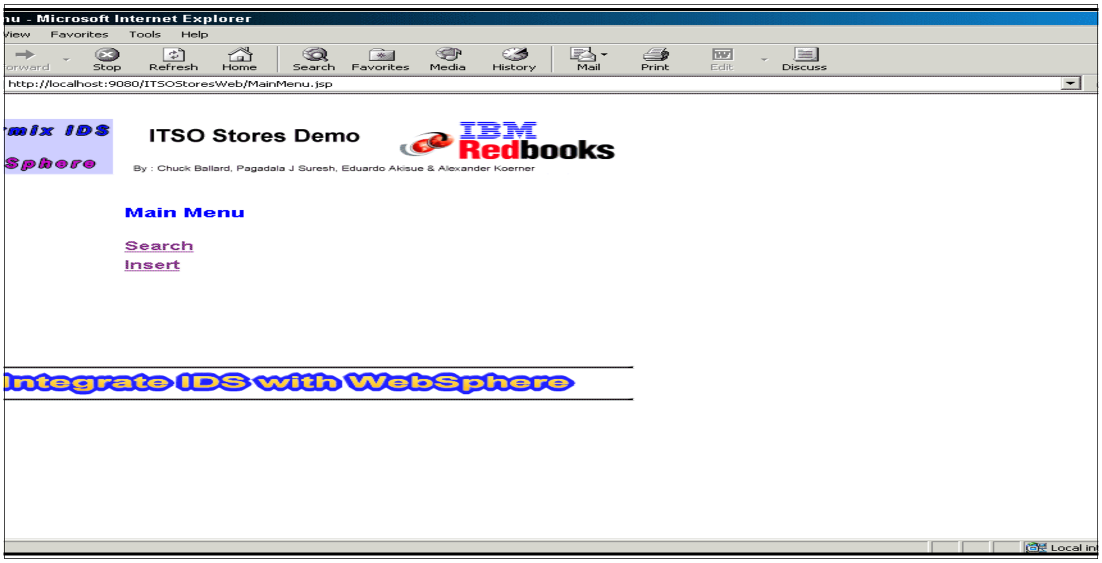
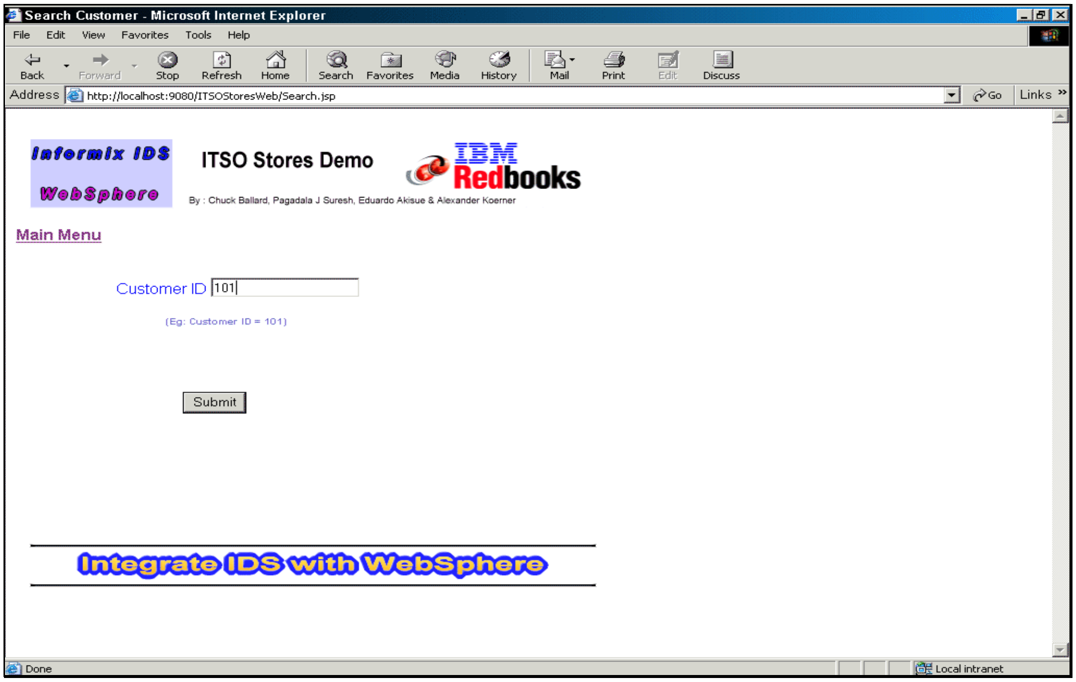

nu - Microsoft Internet Explorer
View Favorites Tools Help
orward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
http://localhost:9080/ITSOStoresWeb/MainMenu.jsp
rmix IDS
Sphere
ITSO Stores Demo
IBM Redbooks
By : Chuck Ballard, Pagadala J Suresh, Eduardo Akisue & Alexander Koerner
Main Menu
Search
Insert
Integrate IDS with WebSphere
Local in

Figure 7-9 Main Menu page for the ITSO Stores Demo

With this application we have the option to search for a particular or to insert a new customer. If we elect to search for a customer, we simply click the word **Search**. The query page will be displayed, as depicted in Figure 7-10.

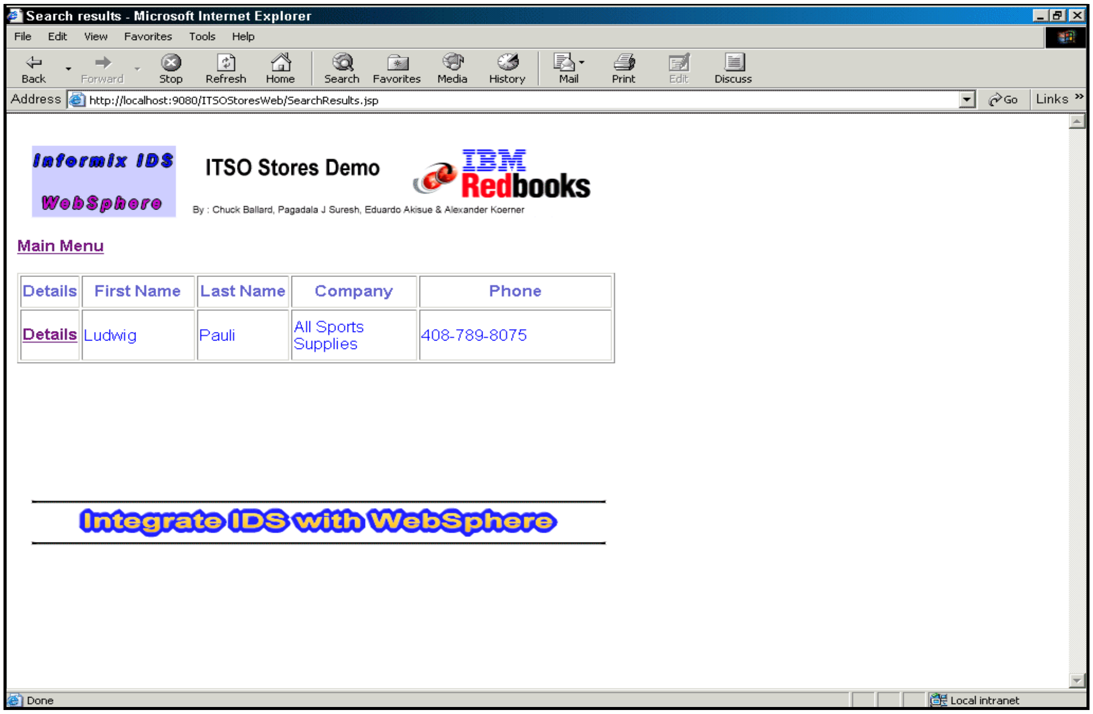

Search Customer - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://localhost:9080/ITSOStoresWeb/Search.jsp
Go Links
Informix IDS
WebSphere
ITSO Stores Demo
IBM Redbooks
By : Chuck Ballard, Pagadala J Suresh, Eduardo Akisue & Alexander Koerner
Main Menu
Submit
Integrate IDS with WebSphere
Done
Local intranet

Customer ID 101

(Eg: Customer ID = 101)

Figure 7-10 Customer Query Page

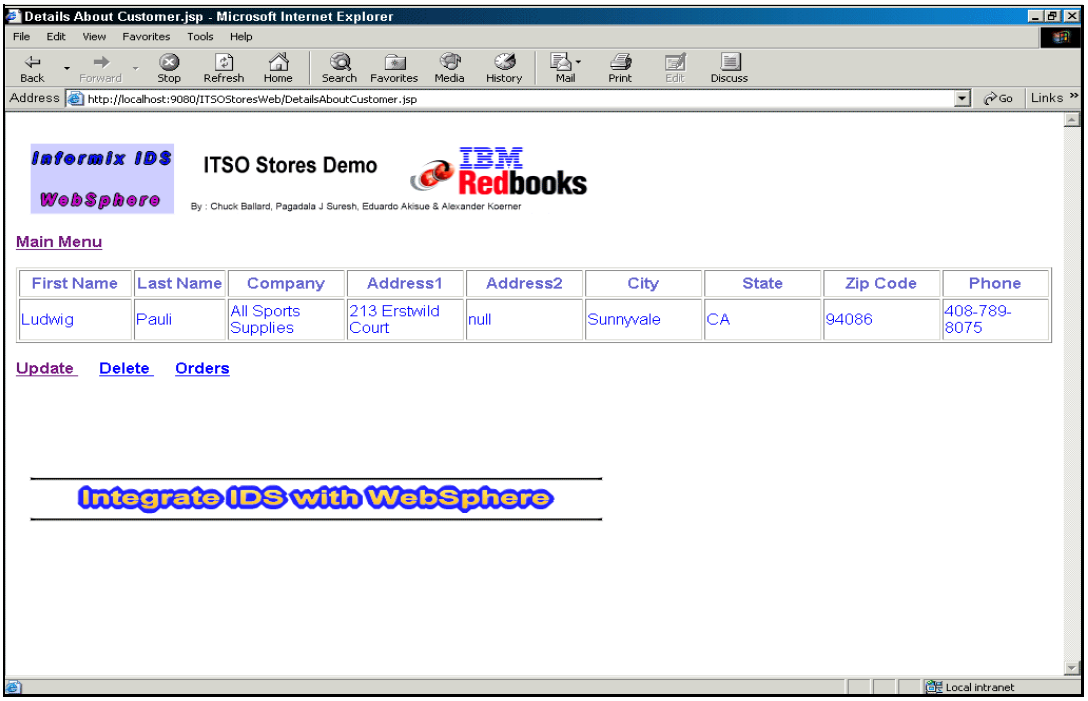
To perform a query, enter a customer number. If the customer number is found, results will be displayed as in Figure 7-11. If not, an error page is displayed.


Search results - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://localhost:9080/ITSOStoresWeb/SearchResults.jsp
Go Links
Informix IDS
WebSphere
ITSO Stores Demo
IBM Redbooks
By : Chuck Ballard, Pagadala J Suresh, Eduardo Akisue & Alexander Koerner
Main Menu
Integrate IDS with WebSphere
Done
Local intranet

Details	First Name	Last Name	Company	Phone
Details	Ludwig	Pauli	All Sports Supplies	408-789-8075

Figure 7-11 Customer Query Results

We now have an option to see more details on this particular customer, by clicking the highlighted word **Details**, shown in Figure 7-12. The Customer Detail Record is displayed.



Details About Customer.jsp - Microsoft Internet Explorer

http://localhost:9080/ITSOStoresWeb/DetailsAboutCustomer.jsp

ITSO Stores Demo

By : Chuck Ballard, Pagadala J Suresh, Eduardo Akisue & Alexander Koerner

Main Menu

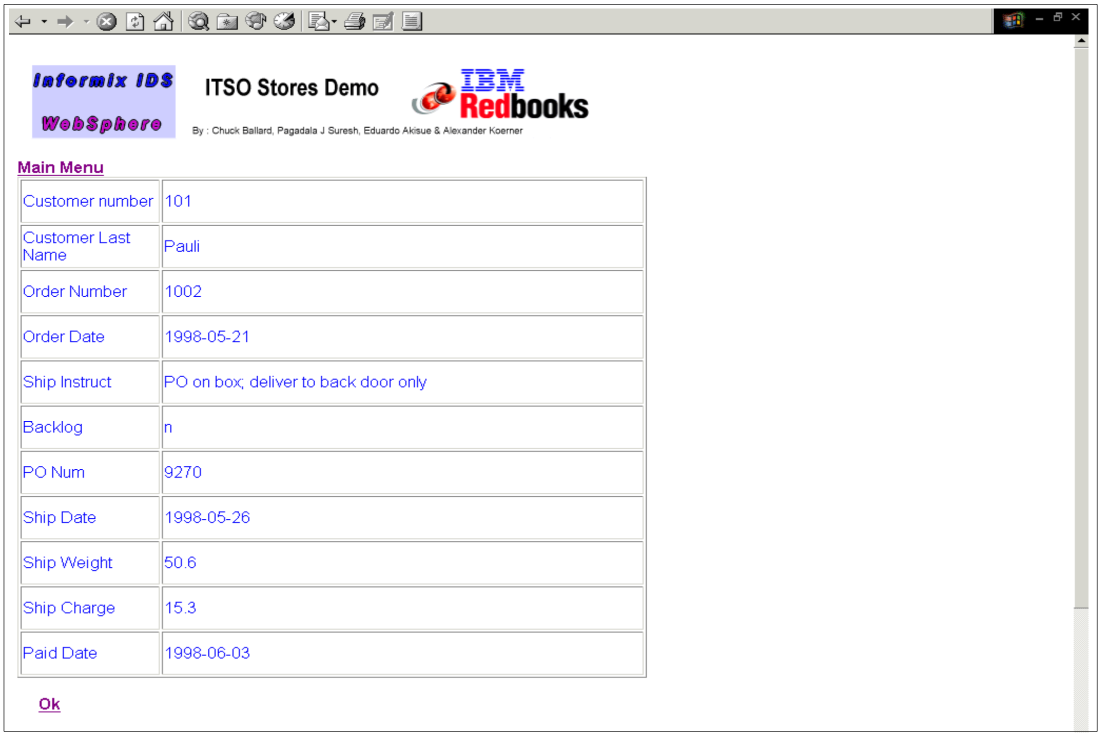
First Name	Last Name	Company	Address1	Address2	City	State	Zip Code	Phone
Ludwig	Pauli	All Sports Supplies	213 Erstwild Court	null	Sunnyvale	CA	94086	408-789-8075

Update Delete Orders

Informix IDS
WebSphere
IBM Redbooks
Integrate IDS with WebSphere
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address
Go
Links
Local intranet

Figure 7-12 Customer Details Record

We can now choose an action to perform on this returned information by clicking the appropriate word. If we click **Orders**, as shown in Figure 7-12, we will retrieve the Customer Order information as depicted in Figure 7-13.



ITSO Stores Demo

By : Chuck Ballard, Pagadala J Suresh, Eduardo Akisue & Alexander Koerner

Main Menu

Customer number	101
Customer Last Name	Pauli
Order Number	1002
Order Date	1998-05-21
Ship Instruct	PO on box; deliver to back door only
Backlog	n
PO Num	9270
Ship Date	1998-05-26
Ship Weight	50.6
Ship Charge	15.3
Paid Date	1998-06-03

Ok

Figure 7-13 Customer Order Detail

This is a simple application, but demonstrates some very robust technology and capabilities. The next sample application will be similar, but uses JMS to demonstrate the capability to implement applications using IDS and WebSphere in a distributed environment.

7.4 A sample application using JMS and IDS

In this section we describe our next sample application, which uses JMS and IDS.

Using JMS with the ITSO Stores sample application

This sample application is designed to demonstrate IDS, WebSphere MQ, and WebSphere Application Server, working together. We have named it ITSOStoresJMS. This sample application is to demonstrate the capability for deploying our sample application, using IDS and WebSphere, in a distributed environment. Distribution can be demonstrated both physically and logically, because the concepts are the same. It is the separation of the system components — in this case, the application, IDS, and WebSphere.

Separation of the components can, and in our example does, require the application to be split into multiple applications to enable them to interact with the appropriate distributed components. In our sample application, there is an application to accept input from the WebSphere user and an application to interact with IDS. There are several ways to implement and support this distribution. WebSphere has a powerful capability to support this, called WebSphere MQ.

However, for our example, we have chosen a simpler way, since we are only working with a sample application. We do not require the very robust capabilities of WebSphere MQ. So, we use Java Message Services (JMS), which can be thought of as a scaled-down version of WebSphere MQ. This is a very simple approach for our sample application, because WebSphere Studio Application Developer and WebSphere Application Server have a JMS Provider embedded in them. Figure 7-14 depicts the design of our sample application environment.

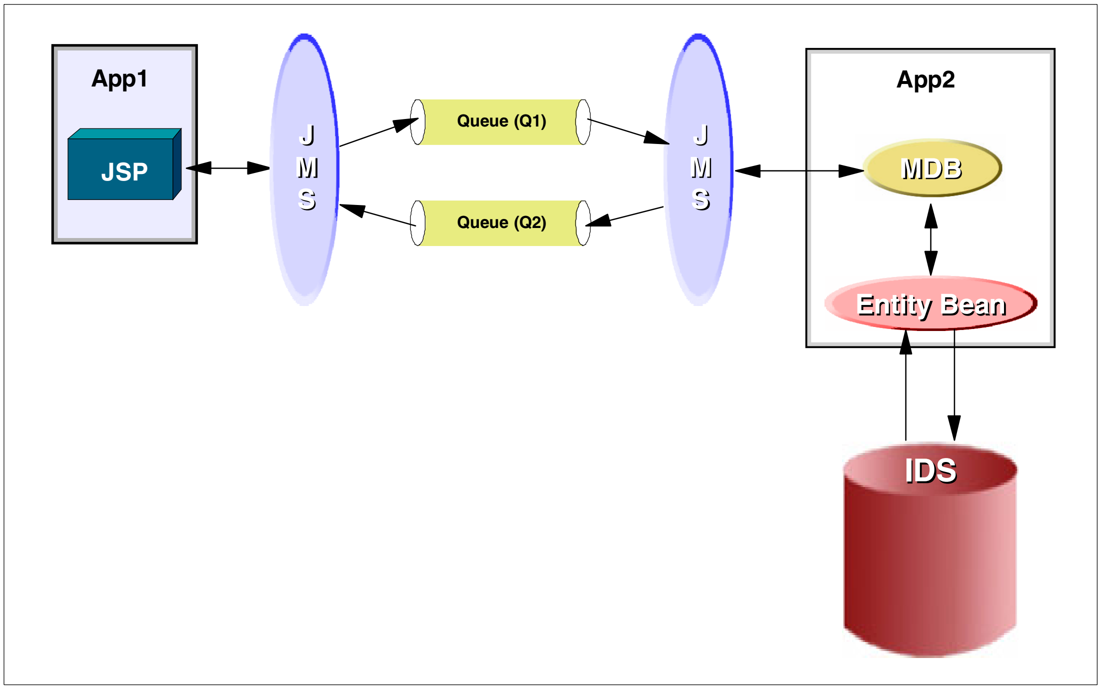


Figure 7-14 Design of ITSO Stores JMS application

As mentioned above, there are two sample applications used, TestJMS (App1) and ITSOStoresJMS (App2). The functionality of App2 includes inserting new customer records into the IDS stores_demo database. The *TestJMS* application, App1, accepts customer detail information from the user, puts it into a JMS queue, and displays the results of the operation to user. ITSOStoresJMS then reads the customer details from queue and inserts a new customer record into IDS.

The two sample applications demonstrate the use of MQ in a distributed environment, using MQ (JMS) as the vehicle for communications between them. Communication between these applications could have been achieved in number of other ways. Here are a few examples:

- **Using RMI:** Using Remote Method Invocation, communication between software components could be established but each component must be aware of the API of the other. If the API of one software component changes, it effects the communication between these components. So communication between software components using RMI is tightly coupled, which may not be desired and is certainly not as flexible. In addition, RMI only supports synchronous communications. And, to communicate, both software components must be up and running (which is not a requirement with WebSphere MQ). So, RMI would only be used when the requirements dictate these particular capabilities.
- **Writing software adapters:** Another way to establish communication between different applications is developing software adapters, which can understand both applications. So this adapter will act a mediator between the applications. But if there is a need to establish a communication between a huge number of applications, it is typically not practical to develop an adapter that can understand them all.
- **Message Oriented Middleware (MOM):** This is an interesting way to communicate between different applications. There are two primary approaches for application communications. In one case the applications must release their own API to the external world, and in the other adapters are written after the applications are developed. Message Oriented Middleware is a type of adapter which is written ahead of the applications. Certain vendors developed message oriented middleware and released their own API. So applications can use these APIs to send and receive messages from the MOM, by using queues. If two applications want to communicate, one will put message into the queue of MOM and the other will read the message from that queue. Here, the communications is through messages. These queues can store the messages if receiving component is not currently available when a message is received. Stored messages are forwarded to receiving application when it becomes available. This is an example of asynchronous communications. WebSphere MQ is the IBM implementation of MOM.
- **Java Messaging Service (JMS):** Message Oriented Middleware is the defacto approach for inter-application communication. However, with multiple vendors implementing their own version of a MOM, support, maintenance, and integration were always issues. As an attempt to standardize the MOM approach. Sun, IBM, and other MOM vendors developed a common messaging API, and it was called JMS.

7.4.1 The ITSOStoresJMS sample application

Coming back to our example, two applications were designed and communications between these components was established using JMS, as depicted in Figure 7-14 on page 158.

A Message Driven Bean (MDB), Container Managed Persistent Bean (CMPB), JSPs, and Servlets are used in this application. Listener ports, queues, and queue connection factory, are configured in the server to enable the application.

The application begins with module *Insert.jsp,* which accepts new customer details from the user and puts them into queue Q1, using the JMS API. The Message Driven Bean (MDB), in App 2, will be listening for the queue, Q1. As soon as there is a message in Q1, the *onMessage method* in the MDB will be executed. The customer details are extracted from Q1 and, with the help of the Container Managed Persistent Bean (CMPB), a new customer record is inserted into customer table, and the customer number of inserted record is placed into queue Q2 as a response. The Java Server Page (JSP) in App1 will be listening for queue Q2, and retrieve the customer number of the newly inserted customer and display it to user.

Downloading the application

The example applications in this redbook were designed to enable you to develop them as an implementation exercise. However, they can also be download directly from our redbook download Web site.

Download the demo sample application file called *SampleApps.Zip* from Appendix D, "Additional material" on page 343. Inside the zipped download file are six folders. Open the folder named *ITSOStoresJMS*. Inside is the actual sample demo applications, called *ITSOStoresJMS.ear and TestJMS.ear. You will use these files later in"Steps to deploy the application in WSAD:" on page 162 and in "Steps to deploy an application in WAS:" on page 163.*

However, to run the applications, you must still install and configure WebSphere Studio Application Developer, WebSphere Application Server, and Informix Dynamic Server. So, there is some preparation work to be done even with this approach.

7.4.2 Preparing the sample application for deployment

In this section we explain how you can prepare the sample application for deployment.

Preparing the sample application for deployment in WSAD

Prerequisites :

The following configuration steps must be finished to enable the sample application to be deployed in WSAD.

Enable the administration client

1. Switch to the Server perspective and select the **WebSphere V5.0 Test Environment**. (**Servers ->WebSphere V5.0 Test Environment**). Right-click **Open**.
2. Select the **Configuration notebook** tab and enable the check box: **Enable administration client**. The Ports Notebook tab will show the Admin host port: 9090.
3. Press **Ctrl+S** to save changes and close the editor using **Ctrl+F4**.
4. Start test server:
   a. If your V5.0 test server has already started, then restart the Web server. To restart, select the **WebSphere V5.0 Test Environment** in the server's view, and right-click **Restart**.
   b. If your server is not started, then right-click **Start** from the pop-up menu.
5. After the server starts, check the WebSphere console log in the server's Console view. You should find following messages:

```
ApplicationMg A WSVR0221I: Application started: adminconsole.
```

Configuring Listener Port LP0

6. Open the Administration Client by using the following URL:

```
http://<I P Address of WSAD>:9090/admin
```

7. In the Navigation pane, select **Servers -> Application Servers.** This displays the properties of the application server in the content pane. Select **your-app-server** from the Application Server's list.

8. In the Additional Properties table, select **Message Listener Service**. This displays the Message Listener Service properties in the content pane.

9. In the Content pane, select **Listener Ports**. This displays a list of the listener ports.

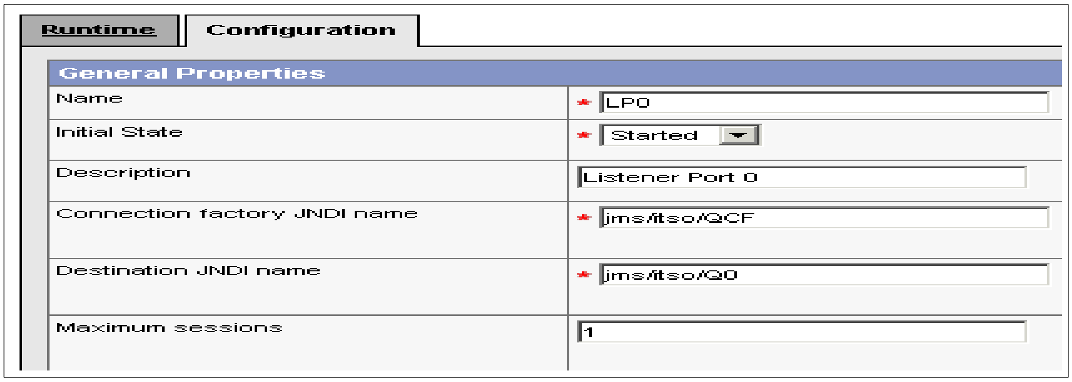
10. In the Content pane, click **New**.

11. Specify appropriate properties for the listener port (refer to Figure 7-15). Enter all required fields:

    a. Name (LP0)
    b. Initial State (Started)
    c. Connection factory JNDI name (jms/itso/QCF)
    d. Destination JNDI name (jms/itso/Q0)

12. Click **OK**.

13. To save your configuration, click **Save** on the task bar of the administrative console window. In the application pane you must click **Save** again.



General Properties	
Name	* LP0
Initial State	* Started
Description	Listener Port 0
Connection factory JNDI name	* jms/itso/QCF
Destination JNDI name	* jms/itso/Q0
Maximum sessions	1

Figure 7-15 Listener Port

Configuring Queue Q0

14. To map queue name to their corresponding JNDI names for the WebSphere JMS provider, select **Resources->WebSphere JMS Provider** in the Navigation tree view.

15. You should find the WebSphereJMSProvider screen with an **Additional Properties** table.

16. Now select the **WebSphere Queue Destinations**, click the **New** button and fill in Name (Q0), JNDI Name (jms/itso/Q0) and select **Node** localhost. Finish the configuration by clicking the **OK** button.

Configuring Queue Connection factory QCF

17. To map queue name to their corresponding JNDI names for the WebSphere JMS provider, select **Resources->WebSphere JMS Provider** in the Navigation tree view.

18. You should find the WebSphereJMSProvider screen with an **Additional Properties** table.

19. Click **WebSphere Queue Connection Factories** and click the **New** button. Provide the Configuration menu with a **Name** (QCF), a **JNDI Name** (jms/itso/QCF), and select **Node** as *localhost*. Click the **OK** button.

Configuring Listener Port LP1

20. Follow“Configuring Listener Port LP0” on page 161, but enter the following values:

    a. **Name** as LP1.
    b. **Connection factory JNDI name** as jms/itso/QCF.
    c. **Destination JNDI name** as jms/itso/Q1.

Configuring Queue Q1

21. Follow the instructions in “Configuring Queue Q0” on page 161, but enter **Name** as *Q1* and **JNDI Name** as *jms/itso/Q1*.

Set JMS server properties

22. We now have to set the JMS server properties. Select of **Server -> Application Server** and select **server1**.
23. In the **Additional Properties** table select **Server Components**. The Server Component table appears. Select **JMS Servers** and you will see the general properties for the **Internal JMS Server**.
24. In section General Properties for the Internal JMS Server, type the **Queue names** as *Q0* and *Q1*, select **Initial State** as *Started* from the drop-down combo box.
25. Click the **OK** button and save your changes to the master configuration.

Configuring a Data source, ITSOStoresDS

26. Refer “Configure Informix Data Source on WebSphere Studio” on page 132.

Steps to deploy the application in WSAD:

27. Import ITSOStoresJMS.ear file and the TestJMS.ear file into the workspace. You should see projects named ITSOStoresJMSEJB and TestJMSWeb.
28. Right-click ITSOStoresJMSWeb project and click **Properties**. Properties of ITSOStoresJMSWeb Wizard will pop up.
29. Select **Java Build Path** from the left pane.
30. In right pane, go to the **Projects** tab and select ITSOStoresJMSEJB.
31. Go to the **Libraries** tab and click the **Add External Jars** button to select ifxjdbc.jar and ifxjdbcx.jar.
32. Add the ITSOStoresJMS and TestJMS applications to the test server.
33. Restart test server and access with application by entering the following URL :

```
http://<ipaddress of test server>:9080/TestJMSWeb/insert.jsp
```

Deploying the application in WAS

Prerequisites :

34. To run our demo **ITSOStoresJMS,** WebSphere Application Server has to be configured for JMS resources. Open WebSphere Application Server, administration client, by using the following URL:

```
http://<ipaddress of application server>:9090/admin
```

35. The remaining steps for the configuration are very similar to configuration steps just presented for WSAD. Follow the steps beginning at Step “In the Navigation pane, select Servers -> Application Servers. This displays the properties of the application server in the content pane. Select your-app-server from the Application Server’s list.” on page 161, through Step “Click the OK button and save your changes to the master configuration.” on page 162.

Configuring Informix JDBC Provider in application server

36. Refer to Section “Configuring the Informix JDBC Provider” on page 114.

Configuring a Data source ITSOStoresDS under Informix JDBC Provider

37. Refer to Section “Configure Informix Data Source on WebSphere Application Server” on page 136

Steps to deploy an application in WAS:

38. Deploy the applications ITSOStoresJMS.ear and TestJMS.ear in ITSOStores as you did with the other sample applications. Refer to section on “Deploy the ITSOStoresDBPages application in WAS” on page 115.
39. Start the application with following URL:

```
http://<ipaddress of application server>:9080/TestJMSWeb/Insert.jsp
```

Congratulations, you have completed deployment of the application ITSOStoresJMS on WebSphere Application Server. We show an example of the execution of that sample application in the next section.

7.4.3 Store and forward mechanism

Now we are going to demonstrate the store and forward mechanism of WebSphere MQ. The store and forward mechanism is a part of asynchronous communications. It enables messages to be sent to users that are currently not available. Then when the user becomes available, the messages are automatically forwarded. This is just one scenario where store and forward mechanism can be used. This mechanism is extensively used in managing transactions.

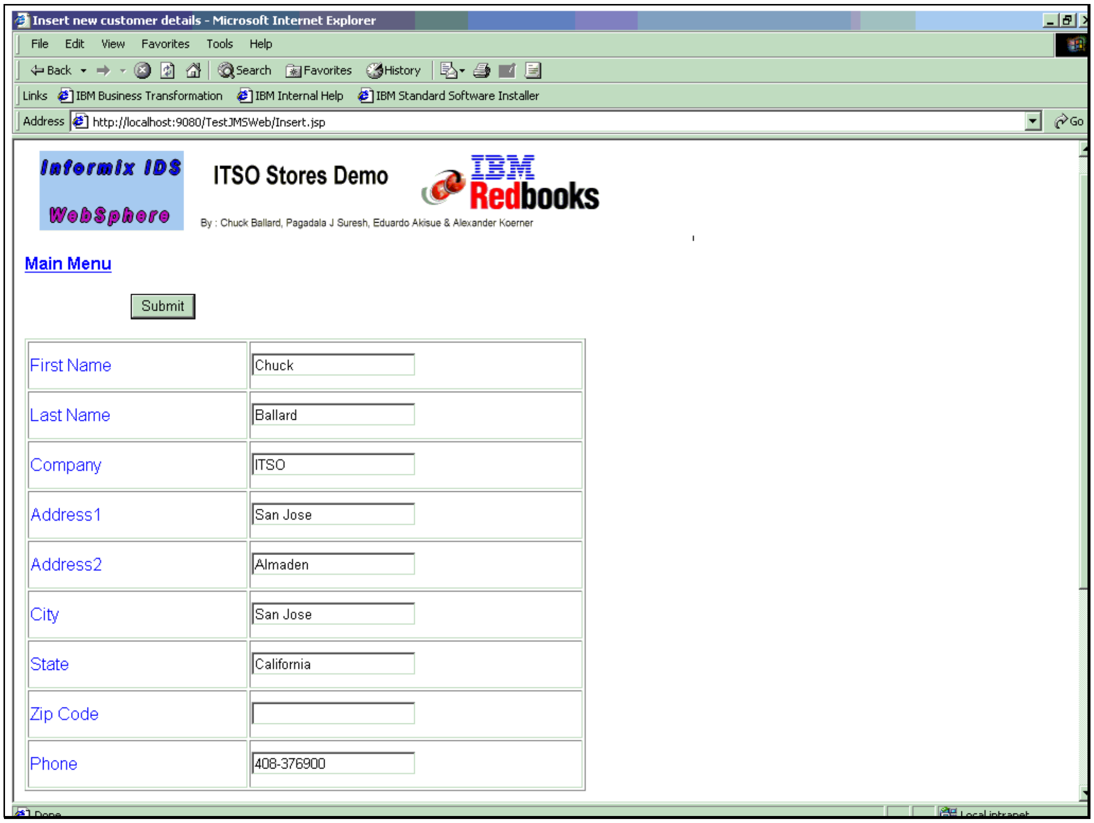
Using our demo, we can demonstrate the store and forward mechanism. The TestJMS application has to collect customer information from user and send it to the application ITSOStoresJMS. But what will happen to the data sent by TestJMS, if the ITSOStoresJMS application is not available to receive it. If you are not using a MOM approach to communications, the process will not complete. With a MOM approach, such as WebSphere MQ or JMS, the data will be stored in a queue until the ITSOStoresJMS application becomes available and formally receives the data. MQ guarantees the delivery of the data, whenever the receiving application becomes available.

The store and forward mechanism of WebSphere MQ

These steps can be used to demonstrate the store and forward mechanism of MQ. In our example we actually used JMS, but the process is the same since JMS is based on MQ:

1. Stop the application ITSOStoresJMS in WebSphere Application Server using the administrative console.
2. Access the insert page of the TestJMS application, as depicted in Figure 7-16, by entering the following URL in your browser:

```
http://<ipaddress of Application Server>:9080/TestJMSWeb/insert.jsp
```

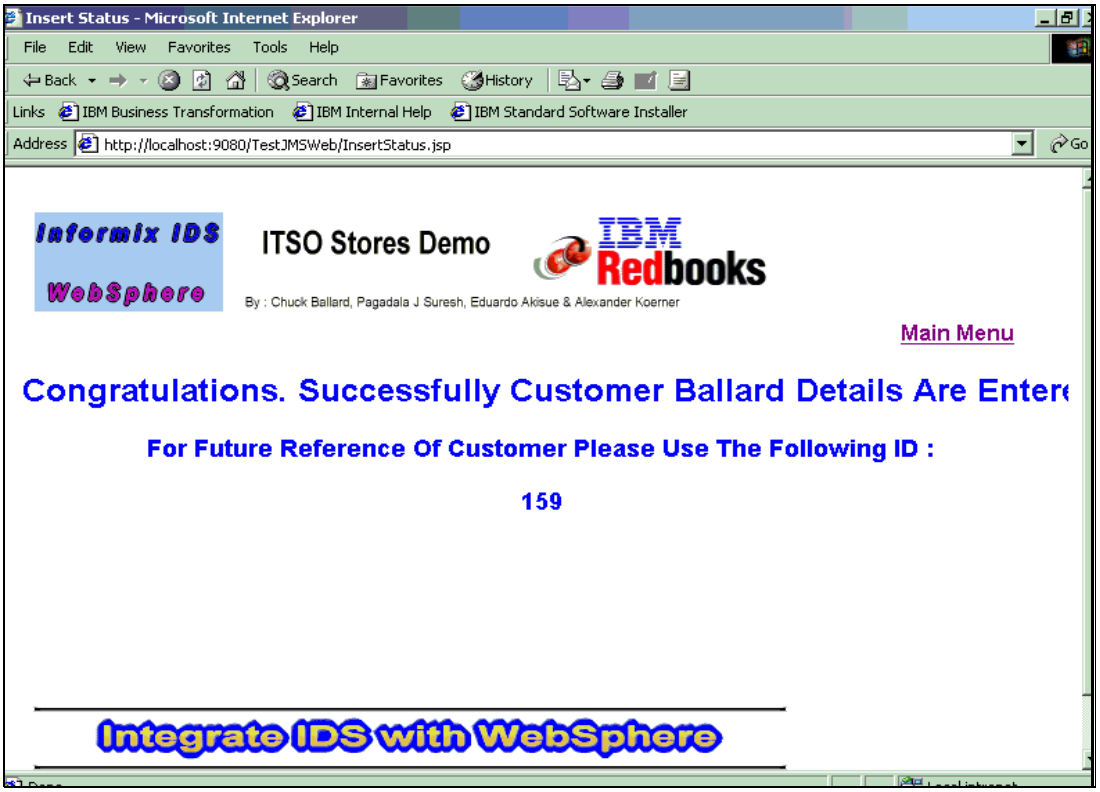

Insert new customer details - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer
Address http://localhost:9080/TestJMSWeb/Insert.jsp
Go
Informix IDS
WebSphere
ITSO Stores Demo
IBM Redbooks
By : Chuck Ballard, Pagadala J Suresh, Eduardo Akisue & Alexander Koerner
Main Menu
Submit
Done
Local intranet

First Name	Chuck
Last Name	Ballard
Company	ITSO
Address1	San Jose
Address2	Almaden
City	San Jose
State	California
Zip Code	
Phone	408-376900

Figure 7-16 Insert Page for JMS sample application

3. Enter customer details into the template in Figure 7-16 and click **submit**. This application will put customer details in the JMS queue, but the ITSOStoresJMS application is not available to receive them. In this situation if we use RMI for communication between these applications, it will return a “link failure error”. But the JMS queues store the customer details when the ITSOStoresJMS application is not available, and then forwards it as soon as the ITSOStoresJMS application becomes available.

4. Now start the application ITSOStoresJMS. Confirmation of the insertion of customer details into IDS is displayed on the browser, as depicted in Figure 7-17.


Insert Status - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Links IBM Business Transformation IBM Internal Help IBM Standard Software Installer
Address http://localhost:9080/TestJMSWeb/InsertStatus.jsp
Go
Informix IDS
WebSphere
IBM Redbooks
Main Menu
Integrate IDS with WebSphere

ITSO Stores Demo

By : Chuck Ballard, Pagadala J Suresh, Eduardo Akisue & Alexander Koerner

Congratulations. Successfully Customer Ballard Details Are Entere

For Future Reference Of Customer Please Use The Following ID :

159

Figure 7-17 Insert complete status page for JMS sample application

5. Check the database for the customer record, and you will see that the new customer information record has been inserted. So our demo application completed successfully. And it demonstrates that the store and forward mechanism works, and can support real time transactions.

7.5 Managing transactions

A transaction is the execution of a set of activities as one unit-of-work. This unit-of-work is a set of activities that relate to each other and must be completed together. If any of these activities fail, the entire unit-of-work must be undone. Determining when a transaction begins and ends is called transaction demarcation.

The transaction system itself provides a mechanism to ensure this and it guarantees the four main properties of a transaction: atomicity, consistency, isolation, and durability (ACID).

ACID properties of a transaction

A transaction has the following four properties:

Atomic: A transaction must execute completely or not at all. Every activity in a transactional unit-of-work must execute successfully. If any activity fails, the entire transaction is aborted and all the data changes are rolled back. If all activities execute without an error, the transaction completes and all data changes are committed.

Consistent: A transaction must not leave a system inconsistent after it completes. This means that it guarantees the integrity of the underlying data store. For example, in typical funds transfer operation; there must not be a negative balance in an account after the transfer is complete. The developer should usually enforce consistency.

Isolated: All transactions must be allowed to execute without interference from other processes or transactions. Any intermediate states are transparent to other transactions, allowing multiple transactions to execute serially.

Durable: All data changes committed during a transaction must be written to a persistent data store and should survive hardware or software failures. If a failure occurs, the data can be recovered by using transactional logs.

7.5.1 Java Transaction Service (JTS)

Transactions in EJBs rely on the transaction APIs provided as part of the J2EE specification. JTS defines a low-level API that is meant to be used by the application server provider. JTS is the Java implementation of the OMG CORBA Object Transaction Service (OTS).

> **Note:** EJB developers should use the Java Transaction API (JTA). JTA provides a programming model that developers may leverage for explicit transaction demarcation.

Transactions in J2EE

A J2EE application server makes transactions very easy to use because it masks the complexity of distributed transactions from the developer. It is important to understand the basic terminology of the participants in transaction management:

Resource: Any persistent store on which you can read or write. It could be a database, a JMS queue, or a JCA connector.

Resource manager: Responsible for managing a resource, and is typically a product such as a relational database or message provider.

Transactional object: A component involved in a transaction. In our example, the session bean is a transactional object.

Transaction manager: Component or system responsible for managing the transactional operations. For example, WebSphere is a transaction manager. When an application uses more than one resource manager in one transaction, an external transaction manager is needed. A typical example is reading from a message queue and writing to a database in one transaction. This is a distributed or global transaction. WebSphere can provide that service.

XA protocol: XA is the standard interface between a transaction manager and a resource manager. The transaction manager coordinates a distributed transaction. It typically uses the XA protocol to interact with the database.

Two-phase commit: Resource managers that want to participate in a two-phase commit have to implement the XA protocol, which ensures that the result of a transaction is consistent across all resource managers participating in the transaction. It is used only in distributed transactions. The protocol operates in distinct phases to ultimately commit or abort a transaction:

- **Phase one:** Evaluates the status of each resource manager. The transaction manager checks with each local resource manager whether they are ready to commit the transaction. Each resource manager responds that they are ready or not. A transaction can commit only when all participating resource managers agree during this phase one. This phase is called the prepare phase

- **Phase two:** Concludes the transaction. Based on the response from each resource manager, the transaction manager instructs all resource managers to commit the transaction if all agree, or to roll back the transaction if at least one disagrees. This phase is called the commit phase.

7.5.2 Local and global transactions in WebSphere

When an application uses only one resource during a transaction, for example, when writing to two tables in the same database, then its resource manager can perform the role of transaction manager. This is called local transaction optimization and is transparent to the application. Transactions in this scenario engage in a one-phase-commit (1PC) transaction.

Global transaction is a transaction that uses an external resource manager. A distributed transaction is a transaction over a multi-tier deployment with several transaction participants, such as a database or JCA connection, within the same transaction. A global transaction manager is required to manage distributed transactions, and will coordinate the updates across multiple resources. This is called a two-phase-commit (2PC). The resource manager can only participate in a distributed transaction if it supports the XA protocol. If you want your resources to be part of global 2PC transactions, then you have to ensure that the resources you define in WebSphere associate to XA-compliant drivers. See Figure 7-18.

In this context for **Informix Dynamic Server**

com.informix.jdbcx.IfxConnectionPoolDataSource(1PC)

com.informix.jdbcx.IfxXADataSource(2 PC)

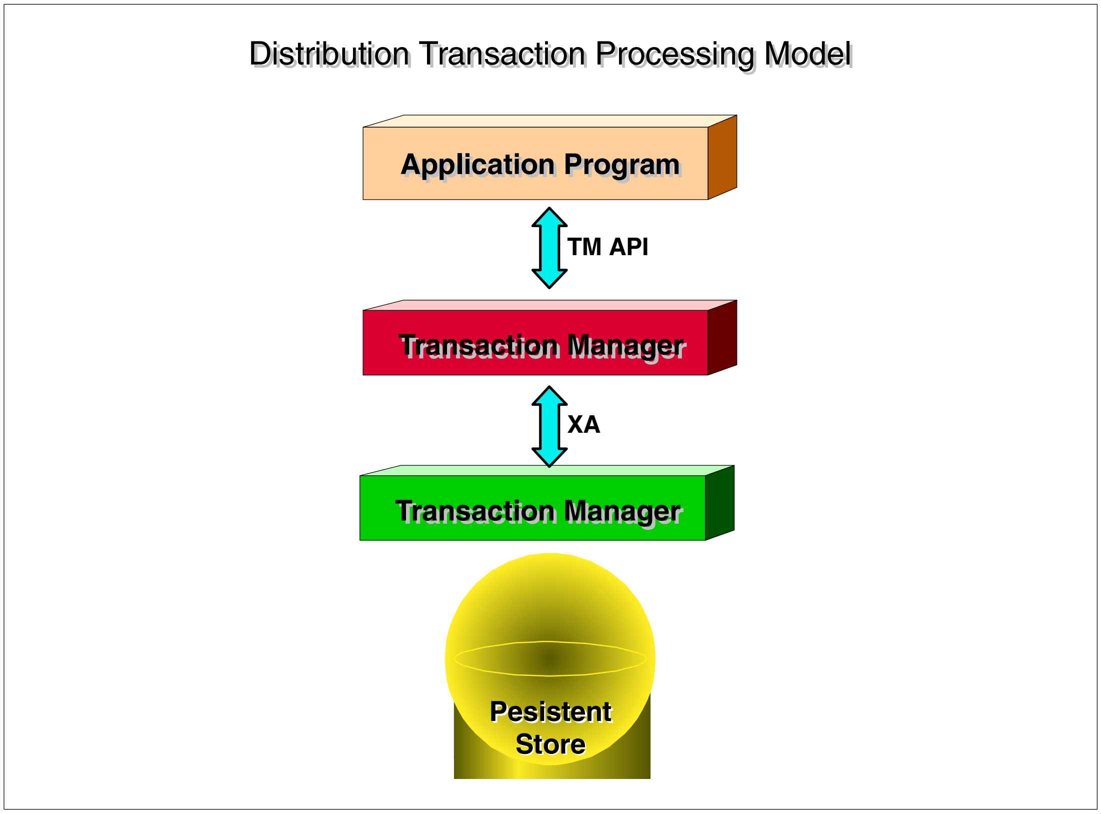


Figure 7-18 Distribution transaction processing

EJB transaction demarcation

Determining when a transaction begins and ends is called transaction demarcation. Bean providers can choose to either control when transactions begin and end programmatically, by using bean-managed transaction (BMT) demarcation, or can delegate this to the container through container-managed transaction demarcation (CMT).

7.5.3 Bean-managed transactions (BMT)

When a developer explicitly manages the bean demarcation levels in the code, they are said to be employing bean-managed transactions (BMT). Bean-managed transactions must declare when transactions start, what behavior is part of the transaction, and when they end.

This is accomplished through the use of the javax.transaction.UserTransaction interface as defined in the Java Transaction API (JTA). This interface provides methods to explicitly begin, commit, and roll back transactions. Only beans that are declared to use bean-managed transactions may use the UserTransaction interface within their code. To do otherwise results in exceptions being by the container. To declare a bean as using bean-managed transactions you must set the <transaction-type> deployment descriptor attribute:
<transaction-type>Bean</transaction-type>

Note: Bean-managed transactions are only available to session and message-driven beans; they are illegal for entity beans. Spanning a transaction across many methods is only allowed in stateful session beans. Stateless session beans, and the onMessage method of message-driven beans, must end or roll back the transaction they start in the method call.

7.5.4 Container-managed transactions (CMT)

BMT can be rather more complicated. A simpler and more elegant solution is to let the container manage the transactions for you. There is no need to write any transaction logic, because this is handled by the container at runtime. The code becomes cleaner, because transaction demarcation is not intertwined in the code. The developer simply has to declare the transactional behavior in the deployment descriptor. One of the most powerful features of EJBs is their ability to employ declarative transaction management.

To declare a bean as using container-managed transactions means setting the <transaction-type> deployment descriptor attribute:
<transaction-type>Container</transaction-type>

Figure 7-19 depicts an EJB object being invoked by a non transactional client. As the client does not have a transactional context, the client creates a new transactional context before dispatching the remote method on EJB object (A). EJB (A) updates database A and then invokes EJB (B), which also updates a database. The processing done by both the EJB objects is with the same transaction context created by the container.

Note: The scenario assumes that EJB (B) has the appropriate transaction attribute, such as TX_REQUIRED, but not TX_NOT_SUPPORTED OR TX_REQUIRES_NEW

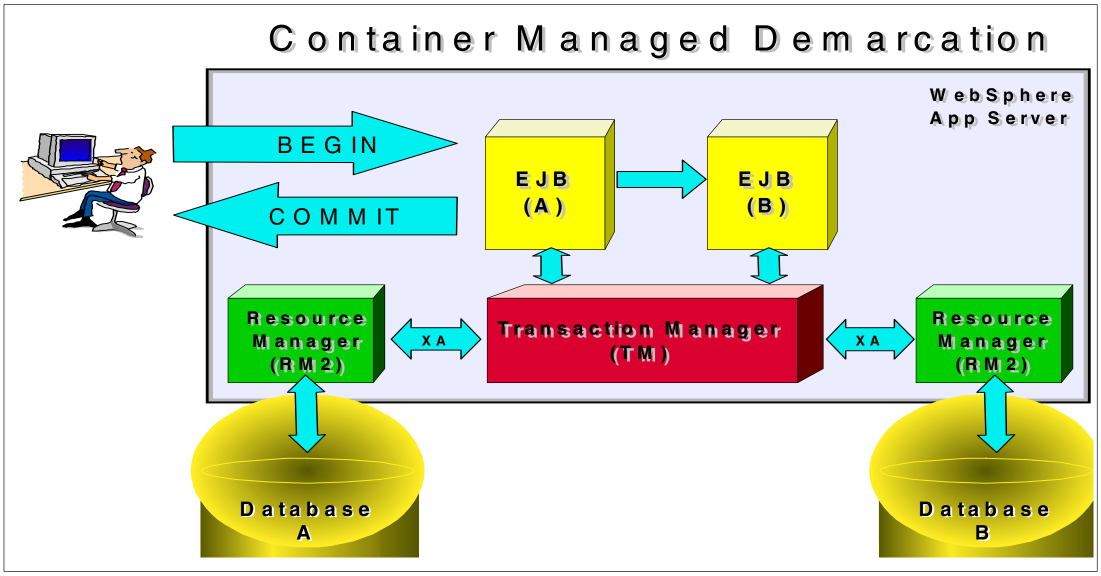


Figure 7-19 Container managed demarcation

In addition, the developer or deployer can declare additional transactional attributes that govern the behavior of the beans in the transactions. The transactional attribute behavior of CMTs can be set at either the bean globally, or the individual methods within a bean. Depending on how the transaction attributes are defined, the container will either start, continue, suspend, ignore, stop, or throw an exception on a transaction when a bean method is executed.

The bean provider is not required to call explicit commit or rollbacks in the code. A transaction starts when a method is called that requires a transaction. The transaction boundary, or scope, is for the length of this method. If the method executes successfully, then the transaction is automatically committed when the method ends.

How does the container know when it should or should not commit? If the bean method (or any other bean method that it subsequently calls) calls the setRollbackOnly method (using the EJBContext object), then the transaction is marked for a rollback. The container guarantees that the transaction is rolled back when completed, but does not force an end of the transaction immediately. The bean developer can use the getRollbackOnly method to determine if a transaction has been marked for rollback, and not continue with the current code sequence if they choose.

There are a number of DBMS features and capabilities of interest when discussing transaction management. The following are a few examples:

- **Database locking strategies:** Relational databases typically use different locking schemes to help isolate access to data. The following are the four types of locks:

  a. **Read locks:** Prevent transactions from changing data read during a transaction until the transaction ends, thus preventing non repeatable reads.

  a. **Write locks:** Used for updates, and prevents other transactions from changing the data until the current transaction is complete. But it does allow dirty reads.

  a. **Exclusive write locks:** Used for updates, and prevents other transactions from reading or changing the data until the current transaction is complete. It prevents dirty reads.

a. **Snapshots:** Some databases provide snapshots, which are frozen views of data. They can prevent dirty reads, non repeatable reads, and phantom reads, but can be problematic because they are not actual data.

- **Deadlocks:** A deadlock occurs when two concurrent transactions place a shared lock on the same resource (table or row) when they read it-then attempt to update the information and commit.
- **Isolation Level:** An isolation level represents a particular locking strategy employed in the database system to improve data consistency with regard to the problems of dirty, non repeatable, and phantom reads. Isolation levels are specified for Enterprise JavaBeans similar to transaction attributes. Since isolation applies only to databases, it is applicable only to enterprise beans using JDBC (either CMP or BMP) to access a database resource. We can either specify a strict isolation or a relaxed isolation. It is a trade-off between concurrency control and performance. That is, a strict isolation level can only be achieved at the expense of performance. The isolation level can be set for a bean, or for individual methods.
- **Limits:** Because locking physically prevents other concurrent transactions from accessing the data, major performance problems may arise. In addition, deadlock of transactions can also occur that may cause stability concerns with the applications. An example of deadlock is when two concurrent transactions are waiting for each other to release a lock. This should be considered when employing an appropriate isolation strategy for your application.

Isolation levels in JDBC

JDBC in particular also deals with transaction and supports its own set of isolation levels. They correspond exactly to the isolation levels supported by enterprise beans. The only exception is the TRANSACTION_NONE isolation level in JDBC, which is not supported by enterprise beans. The equivalent of this isolation level can be achieved by specifying the bean transaction attribute as Never.

Resource access intent

In WebSphere 5.0 the isolation level and read only method level modifiers that could be defined for EJB1.1 are now part of the access intent mechanism of WebSphere. Isolation levels were specific to JDBC, but because the persistence mechanism is now based on J2C resources, a more abstract mechanism was needed. If the underlying resource is a JDBC resource, then the access intent hints will still be translated to JDBC isolation levels under the covers. If the resource is not a relational database, then the access intent will be translated to the mechanism appropriate to that resource.

An access intent policy is a named set of properties (access intents) that governs data access for EJB persistence. You can assign a policy to individual methods on an entity bean's home, remote, or local interfaces during assembly. Access intents are available only within EJB 2.x-compliant modules for entity beans with CMP 2.x and for BMPs.

Access intent enables developers to configure applications so that the EJB container and its agents can make performance optimizations for entity bean access. Entity bean methods are configured with access intent policies at the module level. A policy is acted upon by either the combination of the WebSphere EJB container and persistence manager (for CMP entities) or by BMP entities directly. Note that access intent policies apply to entity beans only. The intent of the access intent mechanism in WebSphere Version 5 is to allow developers to supply the container with optimization hints. The container will use these hints to make decisions about isolation levels, cursor managements, and so forth. The hints are organized into groups called policies. The policies are defined at the module level and applied to individual methods on the bean’s interface (local or remote).

7.5.5 Transactional programming considerations

This section addresses the various programming considerations of transaction management that a bean developer has to consider.

Client-managed transaction: Client-managed transaction refers to the situation in which a client of an EJB manages the transaction demarcation. The client could be a servlet, or another EJB, such as a session bean calling entity beans. For example, this can be a Java servlet that updates records in a database and starts, ends, or rolls back a transaction based on some external criteria. As with bean-managed transactions, we use the UserTransaction interface. Here it can be obtained by a JNDI lookup of java:comp/UserTransaction

New J2EE 1.3: Obtaining a lookup to a user transaction object in previous releases was accomplished using the JNDI context of jta/usertransaction. This is now changed to java:comp/UserTransaction

When the client is demarcating a transaction, it is recommended that the bean methods that the client calls be enabled for container-managed transaction. This simplifies the application and protects the developer from many problems.

Transactions and message-driven beans

Message-driven beans do not have a traditional client, as is the case with session and entity beans. The client exists as a completely disconnected process separated by the JMS provider. Because MDBs do not have a client per se, a message-driven bean cannot participate in a transaction that already exists at the client side. This is not to say that MDBs cannot be transactional, because they can be, but only on the server side in the context under which they execute. Rather, this client restriction implies that the invocation of the MDB cannot be part of a transaction that may have been started on the client side. But clients do not invoke the MDBs directly. They do so by sending a message to a JMS destination, either queue or topic. If the client is utilizing transactions, then the JMS send may be part of the transaction, but not the actual MDB execution itself. This is because the message is only sent when the transaction commits on the client side. Therefore, the MDB would never execute until after the client transaction has actually ended.Having the message listener just stop delivering messages also has its obvious drawbacks. The application, essentially, has just stopped, and messages will continue to queue up, all because of one poison message.

Luckily, using a full-fledged WebSphere MQ implementation for the JMS provider can help in this situation. This is because MQ can be configured to not attempt re-delivery of a message after a certain number of retries of the message within MQ itself, and can put these messages on a special queue destination for temporary holding until a problem can be resolved. For example, WebSphere MQ will attempt to keep a re-delivery count per message, and when the message reaches a pre-determined maximum re-delivery count, say 2, it will put the message on a temporary error queue instead. This would solve the problems of the listener just stopping, the messages queuing up and flooding the queue, or the application becoming unavailable. Of course, a problem still exists that would have to be investigated, but the application continues function for good messages.

Transaction attributes

A transaction attribute is a value associated with a method of session or entity bean's home or component interface or with onMessage method of message driven bean. It specifies how the container must manage transactions for a method when a client invokes via the home/component interface the result of arrival of a JMS message.

Enterprise JavaBeans define the following values for the transaction attribute:

- NotSupported

- Required
- Supports
- RequiresNew
- Mandatory
- Never

7.5.6 General guidelines for using transactions

Here are some guidelines you may find helpful:

- Use long-running read-only transactions to cache static data only.
- Avoid referring to dynamic data in long-running read-only transactions, because the repeatable-read transaction isolation prevents detecting changes committed by other transactions.
- Properly demarcate transaction boundaries and complete transactions quickly.
- The recommended way to manage transactions in EJB is through container-managed demarcation.
- Entity beans must always use container-managed transaction demarcation. Session and message-driven beans can use either container-managed or bean-managed transaction demarcation.
- Bean-managed transaction demarcation should be used when it is the only way to address a problem.
- Stateless session beans must always either commit or roll back a transaction before the business method returns.
- Implement short commit cycles. A commit cycle should not span user think time. A session facade supports this strategy because a method invocation is processed within a single unit of work. Set the transaction demarcation in session beans implicitly through the container (CMT) or as part of the method implementation (BMT).
- The default choice for a transaction attribute is *Required*. Enterprise beans with the *Required* transaction attribute can be easily composed to perform work under the scope of a single JTA transaction.
- The *RequiresNew* transaction attribute is useful when the bean method has to commit its results independent of other transactions, enabling normal transactions to be isolated from critical transactions.
- Session beans that implement the interface SessionSynchronization must have either the Required, RequiresNew, or Mandatory, transaction attribute.
- Message-driven beans may only have a transactional attribute of Required or NotSupported.
- The transaction attributes Mandatory and Never reduce composition of a component by putting constraints on the calling client's transaction context.These attributes can be used when it is necessary to verify the transaction association of the calling client.
- The transaction attribute Supports is not recommended. It has transactional behavior depending on the client association with a transactional context, which is a violation of ACID properties.
- If the message receipt of a message-driven bean is to be part of the transaction, then the MDB must use the Required transactional attribute.
- Try to avoid using Supports and NotSupported, because if the application accesses multiple entities with container-managed persistence in a single high-level business operation, such as a servlet invocation, you may experience unexpected results.

General guidelines for application transaction programming

Here are some guidelines you may find helpful:

- Although not required, application beans should explicitly make the decision to cause a rollback and not leave it to the container. They can do this by calling the EJBContext.setRollbackOnly method. Remember, application exceptions result in a commit unless you explicitly call the setRollbackOnly method.
- Note that setRollbackOnly is only available for container-managed transactions; bean-managed transactions must use the rollback method of the UserTransaction object.
- When calls are returned from downstream EJBs, application beans should check if their transaction has been marked for rollback using the EJBContext.getRollbackOnly method, and act accordingly. They should not just rely on getting this notification as a result of a rollback exception.
- Note that getRollbackOnly is only available for container-managed transactions; bean-managed transactions must call the getStatus method of the UserTransaction object.
- An application can invoke setRollbackonly without necessarily throwing an exception, although this should be avoided.

8

IDS, WebSphere, and XML

In this chapter we give a brief introduction into XML, and discuss the different options one can use with XML in combination with IDS. We then focus on the XML support options in combination with WebSphere Application Developer (WSAD) V5.

We cover these topics:

- XML, a brief overview
- IBM Informix Dynamic Server and XML
- Dynamic XML mapping with WSAD and IDS

8.1 An introduction to XML

In today’s technology, XML is becoming a key piece of software infrastructure. It is an industry standard that supports integration and interoperability by enabling data and document interchange. The main idea is extremely simple. It is a structured document language like HTML and is text based, but the document structure is rigidly enforced, and therefore can be built upon easily. XML documents may use a Document Type Definition (DTD) or an XML Schema.

XML background

XML stands for “eXtensible Markup Language” and is a standard format for describing and exchanging structured data and documents. Unlike HTML, that focuses on the data representation, XML focuses on the data structure which enables easy validation and conversion. Since XML is a meta-language, it allows the definition of your own markup language. See Example 8-1.

Example 8-1 A sample XML document

```
<address>
   <name>
      <title>Mr.</title>
      <first-name>Alexander</first-name>
      <last-name>Koerner</last-name>
   </name>
   <street>20 Oskar-Messter-Street</street>
   <city>Ismaning</city>
   <state>Bavaria</state>
   <zip digits=”5”>85737</zip>
   <country>Germany</country>
</address>
```

Historically speaking, XML is based upon the Standard Generalized Markup Language or SGML. XML is actually a subset of SGM, which has been around for many years. The development of SGML goes back to the late 60s and in based on the work of an IBM employee, Charles Goldfarb and two of his colleagues, Edward Mosher and Raymond Lorie.

In 1998 the World Wide Web Consortium (W3C) approved Version 1.0 of the XML specification. It has since been further developed which has lead to the submission of a working draft of XML 1.1 in April 2002.

One of the success factors for XML in today’s e-business infrastructures has been simplicity and a very broad acceptance in the market.

These are the key features of XML:

- Extensibility through user defined tags
- Separation of the data structure and the presentation format
- Support for deep structures (schema, hierarchies)
- Schema or metadata to describe tags or relationship between tags
- Easy validation and format conversion

The last feature has been an important factor contributing to the current success of XML in the market.

8.1.1 XML usage scenarios

In the late 90s there had been a lot of hype around XML and its usage for potential applications. Some XML vendors even positioned XML as the perfect solution for many existing IT issues far beyond pure data exchange. One could describe this phase as “XML to the rescue”.

As the new millennium began, IT companies took a more realistic approach to the usage of XML and started to focus on the strong areas of XML: Information Exchange, Device Independent Publishing and Document Management.

Let’s take a brief look at these areas...

Information exchange

One could say that the information exchange is probably the most popular usage scenario for XML these days. Until the introduction of XML and related standards (for example, XSL, XSLT etc.) companies were limited to using more costly ways of doing their data interchange such as EDI (Electronic Data Interchange) or EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport). Since XML is freely available and lots of tools have been already developed around XML, even small companies and companies in developing countries can participate in an XML based B2B information flow.

The following list represents typical applications for XML based information exchanges:

- Business to Business data exchange
- Web Services
- Workflow management
- Import and Export of Data

Device independent publishing

Since the introduction of very powerful mobile devices (for example, smart portable phones, PDAs, tablet PCs, and lightweight notebooks) into the market, there has been an increased demand for flexible support of these devices in combination with existing e-business applications. Most of these devices support some kind of internet browser, but typically each has very different kinds of capabilities. Some smart phones support the WAP standard, while others prefer i-mode (or CHTML), some devices have small viewing areas (for example, phones) while others have large screens (for example, tablet PCs). There is a requirement to support information exchange between those different types of devices. A good approach would be to utilize the format conversion capabilities of XML.

Typical device independent publishing XML usages could include, but are not limited to:

- Support for mobile and wireless devices
- On-the-fly format conversions
- More flexible replacement for HTML

Document management

Another very important usage of XML and somehow related to the originating history of XML and SGML, is the management of documents in a structured, but document editor independent way. XML allows easy formatting of even very complex structured documents and related technologies like XPath (XML Path Language) or XQuery (XML Query), allow very sophisticated operations on these documents to support searches and modifications.

Examples of document management centric XML applications could be:

- Easy Format Conversion
- Media Independent Editing

XML document types

Based on the type of applications mentioned in section 8.1.1, we can classify the XML document types listed in Table 8-1.

Table 8-1 XML document types

Document type	Example

Data-centric documents
(structured, regular, typed)
 	
Example: B2B data exchange, product catalog
	`<order>`
	`  <customer>Meyer</customer>`
	`  <position>`
	`    <isbn>1-234-56789-0</isbn>`
	`    <number>2</number>`
	`    <price currency=”Euro”>30.00</price>`
	`  </position>`
	`</order>`

Document-centric documents
(unstructured, irregular, untyped)
 	
Example: Webpage, book, article
	`<content>`
	`XML builds on the principles of two`
	`existing`
	` languages, <emph>HTML</emph> and`
	`<emph>SGML</emph> to create a simple`
	`mechanism ..`
	`The generalized markup concept.`
	`</content>`

Mixed approaches
(data-centric and document centric segments)
 	
Example: Online publications
	`<book>`
	`  <author>Neil Bradley</author>`
	`  <title>XML companion</title>`
	`  <isbn>1-234-56789-0</isbn>`
	`  <content>`
	`    XML builds on the principles of two`
	`existing`
	`    languages, <emph>HTML</emph> and ..`
	`  </content>`
	`</book>`

Throughout this redbook we will focus on the more data-centric and mixed approached documents, since they typically require the dynamic inclusion of data from a database, in our case from IBM Informix Dynamic Server.

XML standards and standard groups

In order to keep XML an open and freely accessible standard to everyone, IBM engages actively in the different leading XML standard groups. In addition the reader will find lots of very useful XML related tools and applications on the IBM alphaworks Web site.

Important XML standards

Here are some of the important XML standards as recommended by W3C:

- **XML Specification 1.0:** Core language syntax, grammar (DTDs)
- **DOM Specification 2.0:** API of parsed objects
- **XSL Specification 1.0:** Transforming and presenting XML
- **XPath Specification 1.0:** Queries, addressing XML docs
- **XHTML Specification 1.0:** HTML in XML form
- **XML Schema:** Big improvements over DTDs

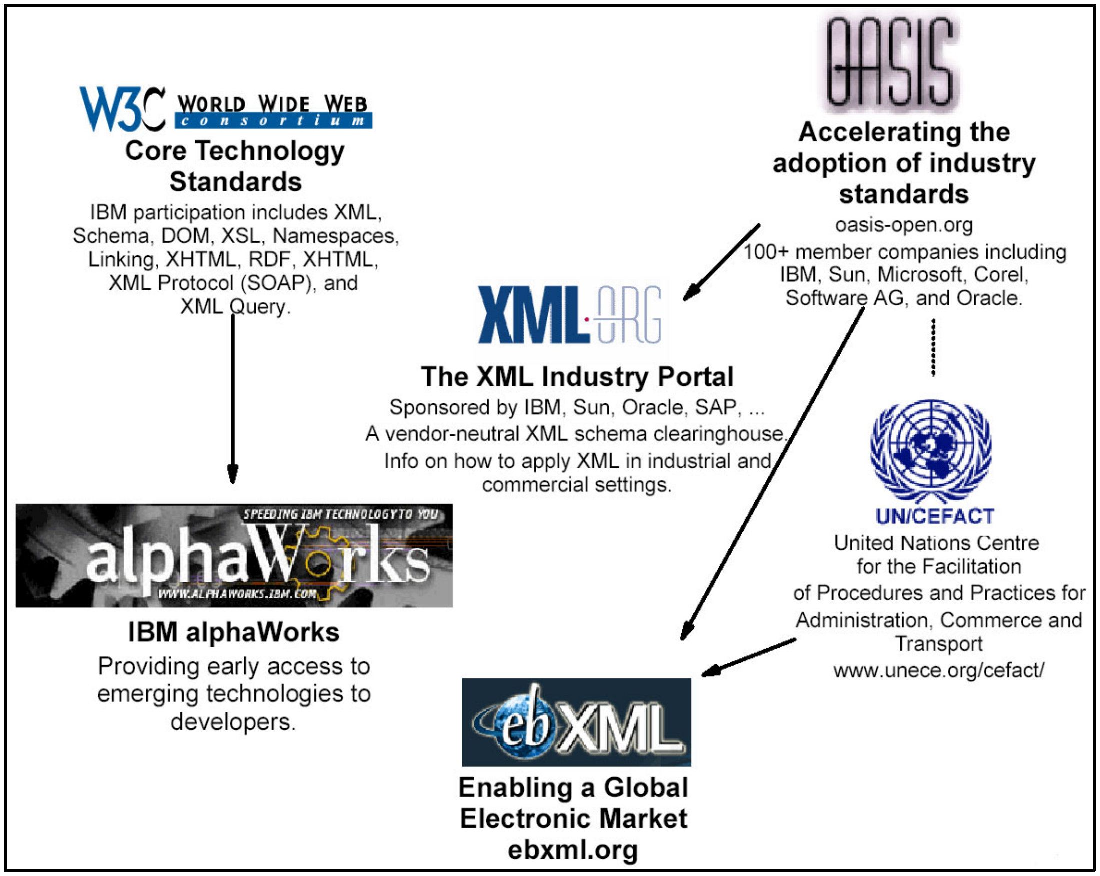

WORLD WIDE WEB consortium
Core Technology Standards
IBM participation includes XML, Schema, DOM, XSL, Namespaces, Linking, XHTML, RDF, XHTML, XML Protocol (SOAP), and XML Query.
OASIS
Accelerating the adoption of industry standards
oasis-open.org
100+ member companies including IBM, Sun, Microsoft, Corel, Software AG, and Oracle.
XML.ORG
The XML Industry Portal
Sponsored by IBM, Sun, Oracle, SAP, ...
A vendor-neutral XML schema clearinghouse.
Info on how to apply XML in industrial and commercial settings.
SPEEDING IBM TECHNOLOGY TO YOU
alphaWorks
WWW.ALPHAWORKS.IBM.COM
IBM alphaWorks
Providing early access to emerging technologies to developers.
UN/CEFACT
United Nations Centre for the Facilitation of Procedures and Practices for Administration, Commerce and Transport
www.unece.org/cefact/
ebXML
Enabling a Global Electronic Market
ebxml.org

Figure 8-1 Interaction between the different XML standard groups

Here are some of the works in progress:

- DOM 3.0 (Document Object Model)
- XML Query: a more powerful XML query mechanism
- XPointer (XML Pointer Language), XLink (XML Linking Language)
- XML Signature (Signature of Web resources and portions of protocol messages), XML Encryption (eEncrypting/decrypting digital content including XML documents)
- XML Protocol (SOAP 1.2)
- WSDL (Web Services Description Language)

These are some other standards:

- SAX 2.0 (defacto standard, not from W3C)
- SOAP 1.1 (defacto standard, now under development @ W3C)

As an interesting side note, XML turned five in February 2003!

8.2 IBM Informix Dynamic Server (IDS) and XML

In this section we start with an overview of functional requirements. We first describe the functionality a typical RDBMS should have in order to successfully support the handling of XML documents, like the ones described in the previous section. As soon as we have laid out the requirements, we zoom into the specific XML support for IDS and discuss the different options a developer might have.

8.2.1 XML support in database systems

XML support in database systems can be defined in four categories: inserting XML, querying XML documents, retrieving XML and XML storage. So let's take a look at the different requirements:

Getting XML in

An XML enabled application has either the need to take an XML document (which could be also an XML fragment) and put it as a whole piece into the database or needs to map the content of an XML document to existing tables in the database.

The first scenario is very common if the documents should be archived as is and later be processed for validation or conversion. Examples of such documents are invoices or any kind of legal documents which could have been part of an intra company workflow, for example. This kind of requirement is also very common for most document-centric XML applications.

In the second scenario the application has to deal with data-centric XML documents and primarily uses XML as a data exchange format, for example, in B2B situations. In this case it is typically important to extract data from an XML document and map it to tables in the (O)RDBMS. This process is often called *shredding* of XML documents. Shredding of XML documents is mostly done by using a middleware layer, sometimes by extensions in the database, or even by built-in database functions.

XML queries

After XML related data have been stored into the database, one would like to be able to query these data or documents for further processing or retrieval.

Due to the hierarchical and semi-structured aspects of XML document layout, its not a very good fit to use plain SQL to query document-centric XML fragments in a database. For this kind of purpose you will find different approaches in the database industry:

- The usage of XPath and/or XQuery to be able to work more XML-like on these documents
- Full text search extensions to the database with some kind of XML support built-in
- Client- or server-side XML parser which operates on XML data stored, for example, in BLOBs

As soon as the components of an XML document have been mapped into regular database tables, one can utilize the power of the existing SQL language again to operate on the elements of the document.

There already have been hybrid approaches to the query problem. For example, by shredding an existing XML document into its hierarchical components, storing it into a database table with some kind of enhanced support for hierarchical structures (Brown, Paul G. “An Object-Relational Approach® to Building a High-Performance XML Repository”, in XML Data Management: Native XML and XML-Enabled Database Systems by Akmal B. Chaudhri, Awais Rashid, Roberto Zicari), and then applying SQL to query the data.

Based on that technique, one could also write a converter which takes XPath syntax, for example, and creates SQL statements to query the shredded elements in the table. In order to keep the original XML document for legal or archival purposes, it could make sense to store an unmodified version in a BLOB in addition to the shredded one. In order to better support the storage of shredded XML documents in IDS, one could use a hierarchical extension to the database through a special node type[1].

[1] http://www7b.boulder.ibm.com/dmdd/zones/informix/library/techarticle/db_node.html

Getting XML out

Based on the initial storage of the XML document, as one piece or shredded, there are different ways to retrieve or re-create the document. In addition there might be the need to create a completely new data-centric XML document from data stored in existing relational tables.

In the case of a complete XML document the process is very simple: just select the document column which could be a CLOB or long VARCHAR (LVARCHAR) data type and send the content to the client application.

If you need to re-construct a document which has been originally shredded into components in a table, you need to construct an SQL statement that will combine the elements back to a complete document or just to an XML fragment. This document reconstruction can be either done in the database server or by using an application server approach (middleware).

One of the most common requirements for constructing XML documents is the dynamic assembly based on data stored in regular (O)RDBMS tables for further processing, especially for the purpose of data exchange and/or Web Services. This kind of processing is sometimes supported in databases through SQL extensions (for example, SQLX), add on databases extensions, or (again) on an application server level.

Storing XML

Although already mentioned in the *Getting XML In* section, let's take a brief look at the storage options.

Whole XML documents will be typically stored BLOB or long VARCHAR data types, while shredded documents will end up in regular tables, maybe enhanced with better support for hierarchical structures.

Some databases do support the concept of an XML data type, if the database does support the extension through user defined types. Those types have associated user defined routines or methods which allow for easy document queries and storage and retrieval.

8.2.2 What's available with IDS

Now, after having discussed the typical requirements for XML support in databases, let's take a closer look at what's available in combination with IBM Informix Dynamic Server. Some of the following approaches work with IDS 7 and IDS 9 while some require functionality only found in IDS 9. To make it easier for the reader to determine which technology is working against which IDS version, we have labelled the following sections accordingly as IDS 7, IDS 9, or IDS 7/9.

IBM WebSphere Application Developer V5 (IDS 7/9)

WebSphere Application Developer V5 (WSAD5) is a very complete environment for any kind of Web application oriented development. One major part in WASD is focused on the development of XML oriented applications and supports such development through different kinds of wizards and libraries.

Of special interest for IBM Informix IDS customers who have a need for dynamic XML mapping against their data stored in tables are two Java libraries: sqltoxml.jar and xmltosql.jar in combination with two wizards: the SQL to XML wizard and the XML to SQL wizard. Additionally, there is the option to generate XML Schemas from DDL statements by using the DDL to Schema wizard.

In 8.3, “Dynamic XML mapping with WSAD V5 and IDS” on page 186 we cover the aspects of the SQL to/from libraries and wizards in greater detail.

Another strong area in WSAD5 is the built-in support for Web services, that are using XML formatted documents for their data exchange and service description. The Web services development with WSAD V5 against IDS is covered in chapter 9.

XML generation and transformation with stored procedures (IDS 9)

The extensibility features in IDS 9, through extended data types and user defined routines (UDR) in C Java and Stored Procedure Language (SPL), allow for easy enhancement of the basic functionality of the underlying database server. Having such flexibility gives you the capability to implement new features in the server without having to wait for the database vendor to provide it.

Following this approach one could very easily implement user defined routines which can return dynamically generated XML, for example, a given table and attributes of this table.

A very good example for such a user defined routine (or stored procedure) can be found on the IBM Informix Developer Zone[2]. In this article, for example, the author, Jacques Roy, describes a function *genxml*, which can be used to generate an XML fragment or, in combination with some helper routines, a complete XML document.

Example 8-2 shows how the genxml function can be called and how the output might look.

Example 8-2 The genxml function (stored procedure)

```
SQL Statement:

SELECT genxml("customer", customer)
FROM customer;


Output of the Statement above:

<customer>
   <customer_num>101<//customer_num>
   <fname>Ludwig         </fname>
   <lname>Pauli          </lname>
   <company>All Sports Supplies </company>
   <address1>213 Erstwild Court  </address1>
   <city>Sunnyvale       </city>
   <state>CA</state>
   <zipcode>94086</zipcode>
   <phone>408-789-8075      </phone>
</customer>
```

In addition to the XML generating function above, there is also an *XSLT (XSL Transformations) DataBlade* available on the IBM Alphaworks Web site[3]. This DataBlade allows the transformation of XML documents by the means of the XSL/XSLT mechanism into a new target format. The XSLT DataBlade stores the XSL documents in LVARCHAR or CLOB data types which allow very complex XSL transformations. It even supports the HTML data type of the Web DataBlade, discussed in the next section.

[2] http://www7b.software.ibm.com/dmdd/zones/informix/library/techarticle/0302roy/0302roy2.html.

[3] http://www.alphaworks.ibm.com/tech/xsltblade

Web DataBlade (IDS 9)

The Web DataBlade is an optional extension of IDS 9 and supports the dynamic generation of any kind of markup language in combination with dynamic data from the IDS 9 database. The initial purpose for the Web DataBlade had been dynamic HTML publishing based on IDS, but it has been extended, for example, to produce XML or SGML documents.

The Web DataBlade utilizes user defined tags within the used markup language, in this case XML. The XML template pages are stored in the database which allows the utilization of IDS 9 server features like replication (ER, HDR), transaction security, and centralized backup.

In combination with the XSLT DataBlade it can be a very powerful way of producing dynamic XML documents, especially for very flexible publishing applications. There is a very detailed article available on the Informix DeveloperZone which covers the combination of the Web DataBlade and the XSLT DataBlade[4].

Although the Web DataBlade is perfect for producing dynamic XML pages, it doesn't solve the issue how to consume XML documents, for example, and map them to database tables. This functionality will be covered in 8.3, "Dynamic XML mapping with WSAD V5 and IDS" on page 186, and the IBM Informix Object Translator is covered under "IBM Informix Object Translator (IDS 7/9)" on page 185.

JAXP support in IBM Informix JDBC Driver (IDS 7/9)

The currently available Java API for programmatically accessing XML documents in the database is called JAXP (Java API for XML Parsing).

The API has the following two subsets:

SAX (Simple API for XML) is an event-driven protocol, that has the programmer provide the callback methods that the XML parser invokes when it analyzes a document.

DOM (Document Object Model) is a random-access protocol, which converts an XML document into a collection of objects in memory that can be manipulated at the programmer's discretion. DOM objects have the data type *Document*.

JAXP also contains a *plugability layer* that standardizes programmatic access to SAX and DOM by providing standard *factory* methods for creating and configuring SAX parsers and creating DOM objects.

IBM Informix extensions to the JDBC API facilitate storage and retrieval of XML data in database columns. The methods used during data storage assist in parsing the XML data, verify that well-formed and valid XML data is stored, and ensure that invalid XML data is rejected. The methods used during data retrieval assist in converting the XML data to DOM objects and to type *Input-Source*, which is the standard input type to both SAX and DOM methods. The IBM Informix extensions are designed to support XML programmers while still providing flexibility regarding which JAXP package the programmer uses.

The IBM Informix JAXP API supports all text related data types in IDS 7 and IDS 9, including BLOB data types such as TEXT (IDS 7) or CLOB (IDS 9) and the LVARCHAR data type.

For more details about how to use the JAXP API in the IBM Informix JDBC driver, you can refer to the *IBM Informix JDBC Driver Programmers Guide*.

[4] http://www7b.software.ibm.com/dmdd/zones/informix/library/techarticle/0303cline/0303cline.html

BYTE/TEXT/BLOB/CLOB/LVARCHAR data types (IDS 7/9)

The usage of Binary Large OBjects (BLOB) for the storage of XML documents in IDS is very straightforward and simple. XML documents can be either stored in the IDS 7 data types BYTE and TEXT or in the IDS 9 data types BLOB, CLOB and LVARCHAR.

These data types can be used in combination with the JAXP API or can be indexed (IDS 9 only!) to allow full text searches across the stored XML documents (Excalibur Text DataBlade and Verity Text Search DataBlade). These data types will be typically used for the archival of document-centric XML documents, for example, to fulfill legal requirements.

IBM Verity Text Search DataBlade (IDS 9)

Although the Verity Text Search (VTS) DataBlade is no longer available to new IBM Informix customers, we want to make customers who have a license of this DataBlade aware that it does support searches in XML documents. In addition to a full text search, as is available with the Excalibur Text DataBlade, the VTS DataBlade allows a search within certain XML tags.

In order to query the XML document in Example 8-3 with the VTS DataBlade and apply the following XML-like Query path “invoice/entries/entry” which identifies all entry elements that are a child of an entry element, which is in turn a child of an invoice element, one has to use the SELECT statement in Example 8-4.

For more details about how to use the VTS DataBlade for XML searches, you can refer to the *IBM Verity Text Search DataBlade Module User’s Guide 1.3*.

Example 8-3 XML document for the SQL query in Example 8-4

```
<?xml version="1.0"?>
<invoicecollection>
   <invoice>
      <customer> Wile E. Coyote, Death Valley, CA </customer>
      <annotation>
      Customer asked that we guarantee return rights
      if these items should fail in desert conditions.
      This was approved by Marty Melliore, general
      manager.
      </annotation>
      <entries n="2">
         <entry quantity="2" total_price="134.00">
            <product maker="ACME" prod_name="screwdriver" price="80.00"/>
         </entry>
         <entry quantity="1" total_price="20.00">
            <product maker="ACME" prod_name="power wrench" price="20.00"/>
         </entry>
      </entries>
   </invoice>
   <invoice>
      <customer> Camp Mertz </customer>
      <entries n="2">
         <entry quantity="2" total_price="32.00">
            <product maker="BSA" prod_name="left-handed smokeshifter" price="16.00"/>
         </entry>
         <entry quantity="1" total_price="13.00">
            <product maker="BSA" prod_name="snipe call" price="13.00"/>
         </entry>
      </entries>
   </invoice>
</invoicecollection>
```

Example 8-4 XML search with the IBM Verity Text Search DataBlade

```
SELECT * FROM table1 WHERE vts_contains
   (text, ' * <in> entry <in> entries <in> invoice);
```

IBM Informix Object Translator (IDS 7/9)

IBM Informix Object Translator (OT) has been part of the former Informix Internet Foundation bundle or has been sold separately to Informix customers. Its still available to IBM Informix customers, although new customers are encouraged to take a look at more complete toolsets such as WSAD V5 that covers most of the OT functionality.

So what does OT do, from an XML/IDS perspective?

In a nutshell: OT allows the dynamic, bi-directional mapping between tables in the database and XML template documents through the creation of a Java intermediate layer. After doing a drag and drop mapping of database tables to intermediate objects and a further drag and drop mapping of these intermediate objects to XML templates, it generates the necessary Java classes that provide a getXml and a setXml method. These methods can then be called from within a servlet engine or a Java application server. OT even supports the SOAP protocol for Web services deployment.

Due to the fact that OT generates Java code, existing OT applications can be easily migrated to a WSAD V5 environment.

IBM Informix Spatial DataBlade 8.20 (IDS 9)

Since version 8.20 of the IBM Informix Spatial DataBlade, this DataBlade has supported the conversion of geometry objects into GML (Geography Markup Language) fragments. GML is based on XML. The conversion of any of geometry object can be accomplished by calling the SE_AsGML() user defined routine in the blade.

Here is a description of the routine and an example:

Typically, you use SE_AsGML() to retrieve the GML representation of a spatial primitive from the server and send it to a client, as in:

```
SELECT SE_AsGML(geomcol) FROM mytable
```

The return type of SE_AsGML() is defined as ST_Geometry to allow GML representations greater than 2 kilobytes to be retrieved by a client application. IBM Informix Dynamic Server automatically casts the output of the SE_AsGML() function to the proper data type for transmission to the client.You can extend the functionality of the IBM Informix Spatial DataBlade module by writing new user-defined routines (UDRs) in C or SPL. You can use SE_AsGML() to convert an ST_Geometry to its GML representation. If you pass the output of SE_AsGML() to another UDR whose function signature requires an LVARCHAR input, you should explicitly cast the return type of SE_AsGML() to LVARCHAR, as in:

```
EXECUTE FUNCTION MySpatialFunc(SE_AsGML(geomcol)::lvarchar)
```

In Example 8-5, the SE_AsGML() function converts the location column of the table mytable into its GML description.

Example 8-5 How to use the SE_AsGML() function

```
CREATE TABLE mytable (id integer, location ST_Point);

INSERT INTO mytable VALUES(1, ST_PointFromText('point (10.02 20.01)', 1000));

SELECT SE_AsGML(location) FROM mytable WHERE id = 1;

<gml:Point srsName="UNKNOWN">
   <gml:coord><gml:X>10.02</gml:X><gml:Y>20.01</gml:Y></gml:coord>
</gml:Point>
```

8.3 Dynamic XML mapping with WSAD V5 and IDS

In the previous sections we discussed the currently available XML features for IBM Informix Dynamic Server outside of the WebSphere environment. Now let's take a closer look into the tools that are provided by WebSphere.

Customers who might have a need for very powerful XML / IDS integration should definitely evaluate the following options related to WSAD and WAS.

Before we focus on the specifics of the XML mapping options in WSAD let's take a brief look at the available XML tools in WSAD in general.

8.3.1 XML tools in WSAD V5

WebSphere Studio provides a comprehensive visual XML development environment. The tool set includes components for building DTDs, XML schemas, XML, and XSL files.

The following XML editor tools are available:

- 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.
- 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 for creating XML instances of an XML schema. You can also use the DTD editor to generate a default HTML form based on the DTDs you create.
- 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.
- 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 your file, you can also validate it. As well, you can associate an XML instance file with the XSL source file you are editing and use that to provide guided editing when defining constructions such as an XPath expression.
- You can use the **XPath expression wizard** 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).

- You can use the **XSL debugging and transformation tool** 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 executed. You can use the views in the XSL Debug perspective to help you debug the XML or XSL files.
- You can use the **XML and SQL query wizard** to create an XML file from the results of an SQL query or 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 of the XML file for use in other applications. Two Java class libraries SQLToXML and XMLToSQL are included so you can uses them in your applications at run time. You can use either the SQL query wizard or SQL query builder to create the SQL queries from which your XML files are generated.
- 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 your 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.
- 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 DB2R XML Extender to either compose XML files from existing DB2 data, or decompose XML files into DB2 data. This tool currently does not work with IDS!

8.3.2 The SQLtoXML and XMLtoSQL framework in WSAD

WSAD includes a very simple to use, but still very powerful framework to support the dynamic mapping of SQL SELECT statements to XML documents and the mapping of XML documents to SQL INSERT/UPDATE/DELETE operations to allow the development of bi-directional XML mapping applications.

You will find the necessary jar files, examples and the documentation for sqltoxml and xmltosql in the directory:

```
<WSAD5_Installdir>\wstools\eclipse\plugins\com.ibm.etools.sqltoxml_5.0.1
```

Detailed help information is also included in the WSAD Help, just type *sqltoxml* into the search field in the main WSAD help window. In addition, for your convenience we have added the sqltoxml and xmltosql class references in Appendix A, “SQLtoXML and XMLtoSQL Java class description” on page 311.

Since SQLtoXML/XMLtoSQL cannot be used without at least a simple servlet or application framework, we will provide you with a simple Java servlet example that can be used as a template for any kind XML mapping requirements in combination with SQLtoXML/XMLtoSQL.

So which basic steps are required to utilize the sqltoxml/xmltosql?

In 8.3.3, “Create a wizard based SQLtoXML sample project” on page 188 we will focus on the SQLtoXML class library and create a sample application by utilizing the wizard driven interface in WSAD.

In 8.3.4, “Enhance the sample project with the XMLtoSQL class library” on page 200 we will then enhance the Java demo servlet by adding the XMLtoSQL functionality manually in the Java demo servlet source code.

The primary classes for generating XML documents from an SQL query are shown in Table 8-2.

Table 8-2 SQLToXML overview

Class	Purpose
SQLToXML	Generate XML from a SQL query. Optionally, generates the corresponding DTD, XML schema and XSL files
QueryProperties	Provides database connection and other parameter settings for calling SQLToXML

The primary classes for mapping XML documents to relational tables are shown in Table 8-3.

Table 8-3 XMLToSQL overview

Class	Purpose
XMLToSQL	Insert, update, or delete rows in a database table using an XML document.
SQLProperties	Provides database connection and other parameter settings for calling XMLToSQL

8.3.3 Create a wizard based SQLtoXML sample project

In this section we are developing a sample project, based on the SQLtoXML framework. The sample application is based on a simple, wizard based SELECT statement and does allow the dynamic XML conversion through XSL stylesheets.

Create a Web project and a connection to the database

You should first start WSAD, then define a new workspace (or choose an existing one), and create a new Web Project, by selecting **File** --> **New** --> **Web Project**.

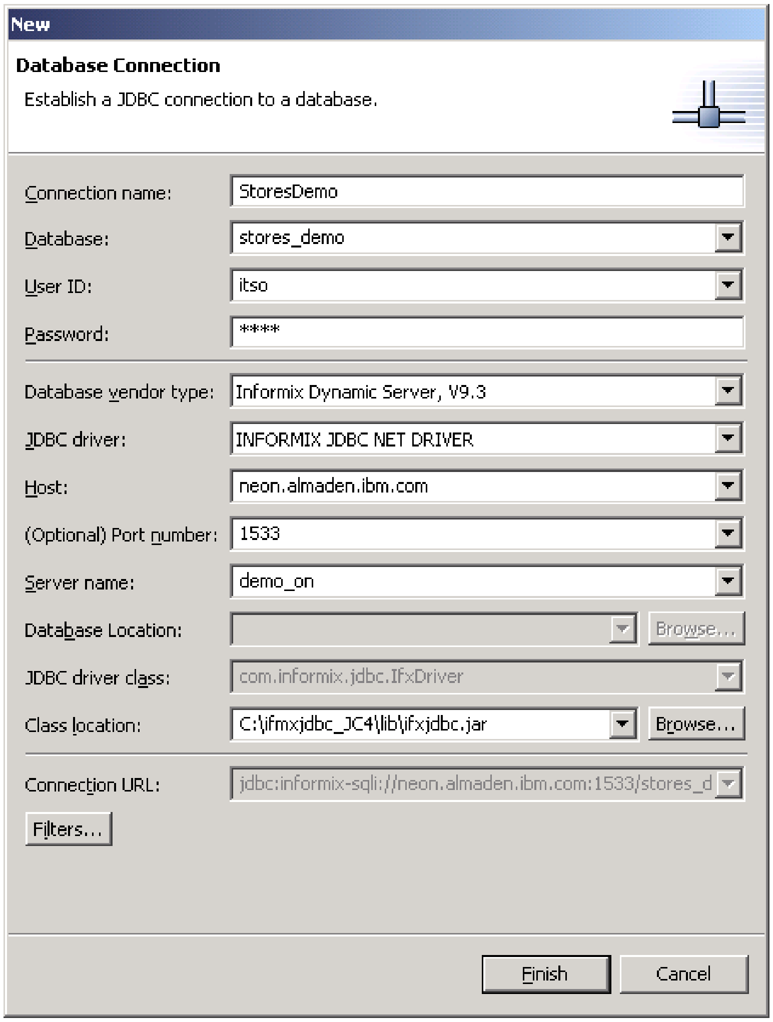
Give the new Web project a name and associate it with a new or existing enterprise application project. In our example we'll name the project **InformixXMLDemo** and the EAR **InformixXMLDemoEAR**. In the J2EE navigator window you should see two folders, one for each object.

In our next step we define a connection to our stores_demo database and create a simple SELECT statement to select all customer data from the demo database.

Switch to the WSAD Data view by selecting **Window** --> **Open Perspective** --> **Data**. Right-click into the DB Servers window and select **New Connection**.

In the New Database Connection fill in the correct connection properties to connect to the stores_demo database. Choose the appropriate Informix Dynamic Server version in the **Database vendor type** field. For IDS 9.4 you should use **"Informix Dynamic Server, V9.3"** and the JDBC driver **Class location** should point to a JDBC driver jar file of version 2.21.JC4 or higher.

So the database connection window should look similar to the one in Figure 8-2.



New

Database Connection

Establish a JDBC connection to a database.

Connection name: StoresDemo

Database: stores_demo

User ID: itso

Password: ****

Database vendor type: Informix Dynamic Server, V9.3

JDBC driver: INFORMIX JDBC NET DRIVER

Host: neon.almaden.ibm.com

(Optional) Port number: 1533

Server name: demo_on

Database Location:

JDBC driver class: com.informix.jdbc.IfxDriver

Class location: C:\ifmxjdbc_JC4\lib\ifxjdbc.jar

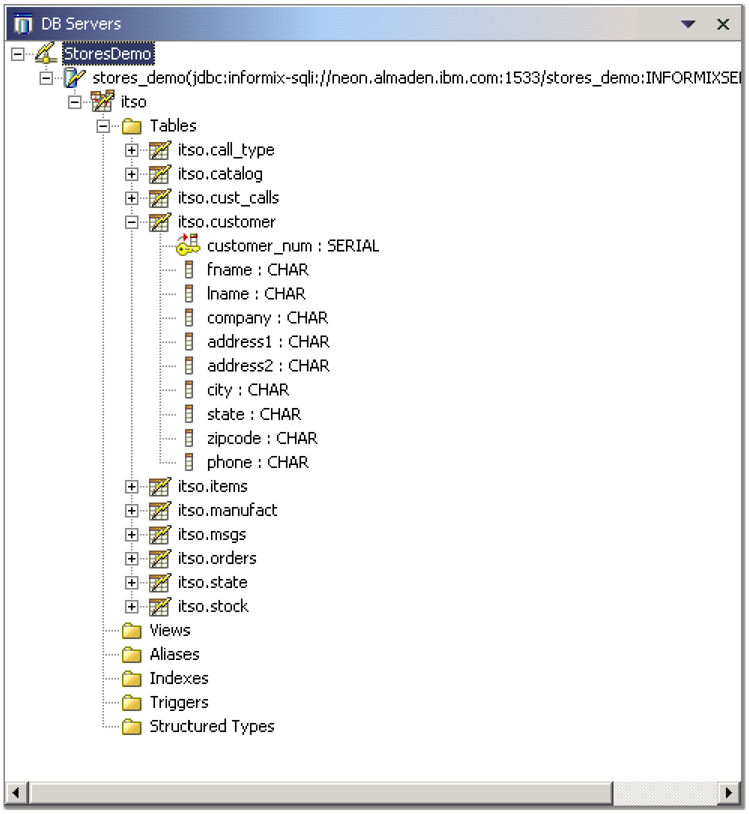
Connection URL: jdbc:informix-sqli://neon.almaden.ibm.com:1533/stores_d

Figure 8-2 New Database Connection wizard

If you might have a need to filter out any kind of unwanted tables or database objects as soon as you connect to IDS, you can do so by setting the filter values via the **Filters** button. A good example would be to filter out the IDS system tables that are normally owned by user informix.

If the database schema import worked without any problems, you should see the imported database objects in the DB Servers window, similar to that shown in Figure 8-3.

Now we need to include the **StoresDemo connection** into our **InformixXMLDemo Web project** by **right-clicking** on the StoresDemo connection name in the DB Servers window and selecting **Import to Folder**.



```
DB Servers
StoresDemo
  stores_demo(jdbc:informix-sqli://neon.almaden.ibm.com:1533/stores_demo:INFORMIXSEI
    itso
      Tables
        itso.call_type
        itso.catalog
        itso.cust_calls
        itso.customer
          customer_num : SERIAL
          fname : CHAR
          lname : CHAR
          company : CHAR
          address1 : CHAR
          address2 : CHAR
          city : CHAR
          state : CHAR
          zipcode : CHAR
          phone : CHAR
        itso.items
        itso.manufact
        itso.msgs
        itso.orders
        itso.state
        itso.stock
      Views
      Aliases
      Indexes
      Triggers
      Structured Types
```

Figure 8-3 Imported database schema in WSAD

As the import folder, choose the Web project InformixXMLDemo via the **Browse** button. Click **Finish** and answer **Yes to** the question as to whether the database folder should be created.

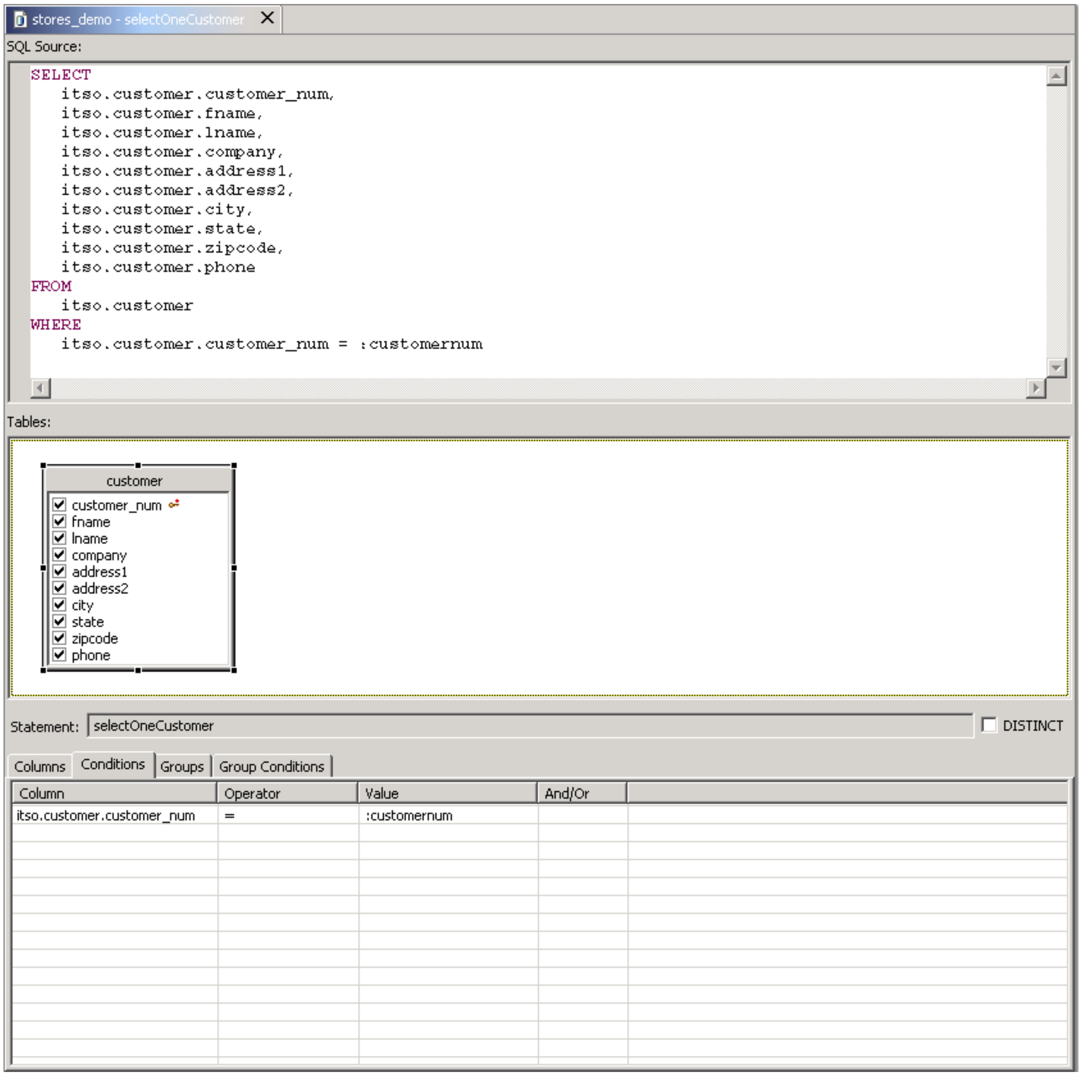
Define a SELECT statement by utilizing the Query Builder

In the Data Definition window, navigate to the **InformixXMLDemo** --> **Web Content** --> **WEB-INF** --> **databases** --> **stores_demo** --> **Statements** folder. Right-click the **Statements** folder and choose **New** --> **Select Statement**. Name the Statement, for example, selectOneCustomer and click **OK**.

Now you should see the interactive query builder. In the tables window, you need to select (by right-clicking) the tables you would like to include in the query. In our demo we select only the table **itso.customer** as we would like to show the complete customer information. Since we would like to include all attributes from the customer table, select all customer attributes in the table attribute check boxes.

We want to select only one customer, therefore we need to define a WHERE condition. To do this, select the **Conditions tab** in the lower window of the query builder. Select itso.customer.customer_num as the column, and choose = as the operator. We also need to provide a host variable, which acts as a placeholder for different customer_num values later in the process. Let’s name the host variable **:customernum** (the colon is important!)

So the query builder windows should look like the one in Figure 8-4.



SQL Source:

```
SELECT
   itso.customer.customer_num,
   itso.customer.fname,
   itso.customer.lname,
   itso.customer.company,
   itso.customer.address1,
   itso.customer.address2,
   itso.customer.city,
   itso.customer.state,
   itso.customer.zipcode,
   itso.customer.phone
FROM
   itso.customer
WHERE
   itso.customer.customer_num = :customernum
```

Tables:

customer

- ☑ customer_num
- ☑ fname
- ☑ lname
- ☑ company
- ☑ address1
- ☑ address2
- ☑ city
- ☑ state
- ☑ zipcode
- ☑ phone

Statement: selectOneCustomer ☐ DISTINCT

Column	Operator	Value	And/Or
itso.customer.customer_num	=	:customernum	

Figure 8-4 The WSAD interactive Query builder

Now save your statement into the Web project by selecting **File --> Save stores_demo - selectOneCustomer**.

Now we have everything in place to start with the XML template generation process.

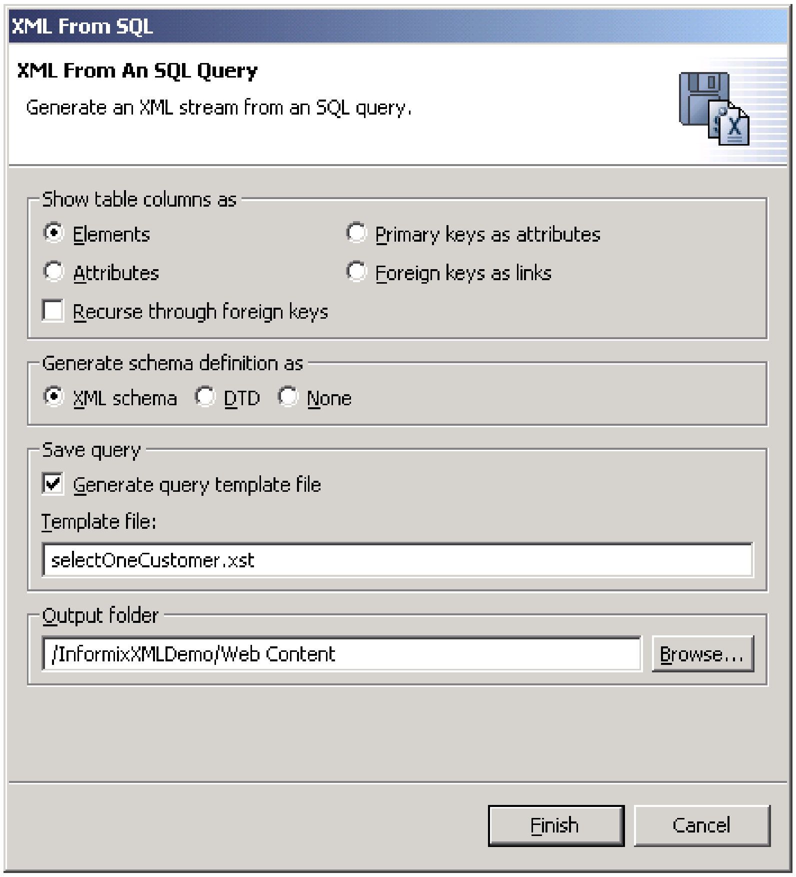
Generate the SQLtoXML template file (.xst)

While still being in the Data Definition window in the Data view, right-click the selectOneCustomer statement. In the popup menu select the **Generate new XML** option. In the XML from SQL wizard window choose the appropriate options. In our example we select the Show table columns as Elements and the Generate schema definition as XML Schema options. Make sure that Recurse through foreign keys has been deselected[5].

Also select the Generate query template file option and name the file selectOneCustomer.xst and the Output folder /InformixXMLDemo/Web Content. By using this folder name, the servlet will have easy access to the XML template file during runtime.

[5] There is currently an issue in the sqltoxml class library that does not support the recursive option in combination with IDS

So the XML from SQL wizard window should look like Figure 8-5.



XML From SQL

XML From An SQL Query

Generate an XML stream from an SQL query.

Show table columns as

● Elements ○ Primary keys as attributes

○ Attributes ○ Foreign keys as links

□ Recurse through foreign keys

Generate schema definition as

● XML schema ○ DTD ○ None

Save query

☑ Generate query template file

Template file: selectOneCustomer.xst

Output folder: /InformixXMLDemo/Web Content

Figure 8-5 XML from SQL wizard

Now click **Finish** to start the XML template file generation. The wizard will ask you for a value for the defined host variable, in our example **:customerid**. If you might encounter a **setQueryTimeout() not supported** error message (depending on the options you might have choosen in the XML from SQL wizard), you can refer to the **Tip** on page 193.

In the selected output folder you should see five different files, all with the selectOneCustomer prefix. Based on the supplied host variable value you will get the query template file (.xst), an XML file with the query results (.xml), an XML Schema (.xsd) or DTD file (.dtd), an default XSL stylesheet for a potential XML to HTML conversion of the query results and an already converted HTML file.

At this time you might want to take a look at the generated XML file to get familiar with the SQLtoXML XML file structure (Example 8-6 on page 193)

Tip:

The sqltoxml class library utilizes the Statement.setQueryTimeout(int) method in the JDBC driver.

The IBM Informix JDBC driver 2.21.JC4 does support this feature, but only in combination with a JRE 1.4 or higher. Since the JRE in WSAD5 is based on version 1.3.1 (for development and internal test deployment) you should apply the following steps as temporary workaround until WSAD supports JRE 1.4:

1. Add the ’-D java.version=1.4.0’ property to the command line options of the WSAD startup command.
2. Add the java.version=1.4.0 property to the Environment section of the internal Test application server configuration (see also Section , “Create a DADX Web service based on the generated DADX file” on page 226)

The root element is labelled **SQLResult** (which cannot be changed) and the element names are based upon the column names or aliases in the SELECT statement. This XML format can be very easily converted in any other XML format by applying an XSL stylesheet in combination with an XSLT processor. Since SQLtoXML already generates a default XSL sheet for an optional XML to HTML conversion, HTML support can be activated by simply applying the default XSL stylesheet to the raw XML document(s). See Example 8-6.

Example 8-6 XML format, generated by SQLtoXML

```
<?xml version="1.0" encoding="UTF-8"?>
<SQLResult xmlns="http://www.ibm.com/customer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/customer
   selectOneCustomer.xsd">
    <customer>
        <customer_num>104</customer_num>
        <fname>Anthony</fname>
        <lname>Higgins</lname>
        <company>Play Ball!</company>
        <address1>East Shopping Cntr.</address1>
        <address2>422 Bay Road</address2>
        <city>Redwood City</city>
        <state>CA</state>
        <zipcode>94026</zipcode>
        <phone>415-368-1100</phone>
    </customer>
</SQLResult>
```

The generated Query template file (*.xst) contains all the necessary information for the SQLtoXML class to be able to connect to the database and execute the query. Since the .xst file is also encoded in an XML format, for example, it can be automatically generated by another application or converted from/to any other format if necessary.

Since we have now the query template file, we need to continue with the development of our Java servlet. See Example 8-7.

Example 8-7 Format of the generated .xst file

```
<?xml version="1.0" encoding="UTF-8"?>
<SQLGENERATEINFORMATION>
  <DATABASEINFORMATION>
```

```
    <LOGINID>itso</LOGINID>
    <PASSWORD><![CDATA[itso]]></PASSWORD>
    <JDBCDRIVER>com.informix.jdbc.IfxDriver</JDBCDRIVER>

<JDBCSERVER>jdbc:informix-sqli://neon.almaden.ibm.com:1533/stores_demo:INFORMIXSERVER=demo_
on;</JDBCSERVER>
  </DATABASEINFORMATION>
  <STATEMENT>
    <![CDATA[ SELECT itso.customer.customer_num, itso.customer.fname, itso.customer.lname,
itso.customer.company, itso.customer.address1, itso.customer.address2, itso.customer.city,
itso.customer.state, itso.customer.zipcode, itso.customer.phone FROM itso.customer WHERE
itso.customer.customer_num = :customernum  ]]>
  </STATEMENT>
  <OPTIONS>
    <FORMATOPTION>GENERATE_AS_ELEMENTS</FORMATOPTION>
    <RECURSE>FALSE</RECURSE>
  </OPTIONS>
</SQLGENERATEINFORMATION>
```

Import necessary class libraries into the Web project

Now we need to import the SQLtoXML and XMLtoSQL class libraries into our Web project to make them available to our Java servlet. Initially we'll only need the sqltoxml.jar file, but for our convenience we will import the xmltosql.jar file too.

To import the jar files select **File** --> **Import** --> **File system** and then click the **Next** button. In the director, select the sqltomxml jar file directory (<WSAD_Install_Dir>\wstools\eclipse\plugins\com.ibm.etools.sqltoxml_5.0.1\jars). Then select the sqltoxml.jar and the xmltosql.jar file. As the destination import folder, select **InformixXMLDemo/Web Content/WEB-INF/lib.**

Now we have to do the same for the IBM Informix JDBC driver: Select **File** --> **Import** --> **File system** and then click the **Next** button. In the directory, select the ifxjdbc.jar file directory (<Informix_JDBC_2.21JC4_InstallDir>\lib\ifxjdbc.jar). As the destination import folder, select again **InformixXMLDemo/Web Content/WEB-INF/lib.**

Create the demo servlet Java code

To create the Java servlet we first start with an "empty" Java servlet template which we create by following these steps:

1. Switch to the J2EE perspective by selecting **Window** --> **Open Perspective** --> **J2EE**
2. In the **J2EE Navigator** window, right-click the **Java Source** folder
3. Select **New** --> **Servlet**
4. Choose a class name. For our example choose: InformixSQLtoXMLServlet
5. Click the **Finish** button

You should see the generated InformixSQLtoXMLServlet.java file in the Java Source folder. If it is not already opened in an editor window, double-click it.

Now we need to fill in some Java for a working Java demo servlet.

Let's start with the import section. You need to replace the import section with the code from Example 8-8.

Example 8-8 Import section of the demo servlet

```
import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

import java.util.*;
import java.net.URLEncoder;

import java.sql.Connection;

import com.ibm.etools.sqltoxml.*;
import com.ibm.etools.xmltosql.*;

import org.xml.sax.SAXException;

import javax.xml.transform.TransformerFactory;
import javax.xml.transform.Transformer;
import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.stream.StreamResult;
```

In the next step we need to fill in the code (Example 8-9) for the doGet() method of the InformixSQLtoXMLServlet. The code in this example is kept already very flexible to not only allow the execution of predefined .xst files, but also the execution of SQL queries dynamically.

Since the servlet will be called from an HTML page we need to be able to handle the required parameters for a later execution. The servlet either accepts an .xst template file or a query string. As soon as an .xst file is supplied it will override any supplied query string. Since the .xst file also includes the database connect information, that information will override the hardcoded connection information in the doGet() method.

The demo servlet also already supports the possible translation of the generated XML format into another format by applying an optional XSL stylesheet. If there is no supplied XSL sheet, the servlet will only return the raw XML, as generated by the sqltoxml class library.

Here is a description of how to use the sqltoxml class, step by step:

1. Either set the QueryProperties manually or load them from a pre-defined .xst file
2. Create a new SQLToXML object and initialize it with the QueryProperties
3. Supply any host variables by using the SQLToXML.setParameters() method (the host variables need to be a comma separated list, in the order of their occurrence)
4. Create a new PrintWriter object which will contain the results of the XML mapping
5. Call the SQLToXML.execute() method to execute the SQL Query and return the XML document
6. Optional: You can convert the generated XML output document into any other format by applying an XSL stylesheet through the xmlTransform method.

Example 8-9 The doGet() method

```
   public void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException {

   resp.setContentType("text/xml");
    PrintWriter response = resp.getWriter();
    String query = getParameter(req,"query");
    String params = getParameter(req,"params");
    String xslFile = getParameter(req,"xslfile");
    String xstFile = getParameter(req,"xstfile");


    try
    {
      QueryProperties prop = new QueryProperties();

       prop.setJdbcDriver("com.informix.jdbc.IfxDriver");
       prop.setLoginId("itso");
       prop.setPassword("itso");

prop.setJdbcServer("jdbc:informix-sqli://neon.almaden.ibm.com:1533/stores_demo:INFORMIXSERV
ER=demo_on");
     prop.setFormat("GENERATE_AS_ELEMENTS");
     prop.setRecurse(false);

      if (query != null)
      prop.setStatement(query);

      if (xstFile != null)
       prop.load(getServletContext().getRealPath(xstFile));

      SQLToXML sql2xml = new SQLToXML(prop);

      ByteArrayOutputStream baosXML = new ByteArrayOutputStream();
      PrintWriter xmlWriter = new PrintWriter(baosXML);

      sql2xml.setParameters(params);
      sql2xml.setXMLWriter(xmlWriter);

      sql2xml.execute();

      String xml;

      if (xslFile.length() > 0)
      xml = xmlTransform (baosXML.toString(), xslFile);
          else
      xml = baosXML.toString();

      response.print(xml);
      response.flush();

      xmlWriter.close();
    }
    catch (Exception e)
    {
    error(response, e);
    }
  }
```

Example 8-10 shows the missing getParameter(), error() and xmlTransform() methods which are required for the servlet to be executed.

Example 8-10 Additional Servlet methods

```
private String getParameter(HttpServletRequest req, String param)
  {
    String[] paramValues = null;
    String paramValue = null;

    paramValues = req.getParameterValues(param);

    if (paramValues != null)
      paramValue = paramValues[0];

    return paramValue;
  }


private String xmlTransform(String xml, String xsl)
    throws Exception
  {
    ByteArrayOutputStream baosXML = new ByteArrayOutputStream();

    TransformerFactory tFactory = TransformerFactory.newInstance();
    Transformer transformer =
      tFactory.newTransformer(new
StreamSource(getServletContext().getResource(xsl).toExternalForm()));
    transformer.transform (
                  new StreamSource(new StringReader(xml)),
                  new StreamResult(baosXML));

    String transformedXml = baosXML.toString();
    baosXML.close();

    return transformedXml;
  }

protected void error(PrintWriter writer, Exception e)
   {
    writer.println("<html>");
    writer.println("<head><title>Error</title></head>");
    writer.println("<body>");
    writer.println("<h2> " + e + "</h2>");
    e.printStackTrace(writer);
    writer.println("</body>");
    writer.println("</html>");
   }
```

Create a simple HTML page to call the demo servlet

In order to be able to call the servlet, we need a write a simple HTML page. Let's start with an empty page first:

1. While being still in the J2EE Navigator view, right-click the Web Content folder. Select **New** --> **HTML/XHTML File**.
2. As the file name enter **InformixSQLtoXML** and then click **Finish**.
3. The template HTML file should open in an editor window. Now switch to the HTML source view.
4. Now replace the content of the file with the HTML code from Example 8-11.

Example 8-11   InformixSQLtoXML.html

```
<HTML>
<HEAD>
<TITLE>InformixSQLtoXML Demo</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">

<P><BR>
<center><FONT SIZE=4 FACE="MS Sans Serif"><B>IBM Informix SQLtoXML /
XMLtoSQL Example</B></FONT></center>
<BR>
<FONT SIZE=2 FACE="MS Sans Serif">Enter a Query and the Params and then
press the <B>Submit</B> button.</FONT>
<BR>
<BR>

<table width=580 border=0 cellpadding=0>
   <tr>
      <td valign=top>
      <FORM name="sqltoxmlsample" METHOD="GET"
         ACTION="/InformixXMLDemo/InformixSQLtoXMLServlet">

      <table border="0">
         <tr>
            <td><B>SQL Query:</B></td>
            <td><input type="Text" name="query"
               VALUE="SELECT customer.customer_num, customer.fname, customer.lname,
customer.company, customer.address1, customer.address2, customer.city, customer.state,
customer.zipcode, customer.phone FROM customer WHERE customer.customer_num = :customernum"
               SIZE=70 MAXLENGTH=150></td>
         </tr>
         <tr>
            <td><B>XST File:</B></td>
            <td><input type="Text" name="xstfile" VALUE="/selectOneCustomer.xst"
               SIZE=70 MAXLENGTH=150></td>
         </tr>
         <tr>
            <td><B>Query Param(s):</B></td>
            <td><input type="Text" name="params" VALUE="103" SIZE=50
               MAXLENGTH=50></td>
         </tr>
         <tr>
            <td><B>XSL File:</B></td>
            <td><input type="Text" name="xslfile"
               VALUE="/ITSO_Customers.xsl" SIZE=50 MAXLENGTH=50></td>
         </tr>
         <tr>
            <td><input type="Submit" name="submit" value="Submit"></td>
            <td></td>
         <tr>
      </table>
      </FORM>

      <FORM name="xmltosqlsample" METHOD="POST"
         ACTION="/InformixXMLDemo/InformixSQLtoXMLServlet">
      <table border="0">
         <tr>
            <td><B>XML Document:</B></td>
            <td><textarea name="xmldoc" rows=20 cols=80>
</textarea></td>
```

```
         </tr>
         <tr>
            <td><input type="Submit" name="submit" value="Submit"></td>
            <td></td>
         <tr>
      </table>
      </FORM>

      </td>
   </tr>
</table>
<BR>
<BR>
</BODY>
</HTML>
```

The InformixSQLtoXML.html file already contains a submission form for the XMLToSQL example which will be discussed in 8.3.4, “Enhance the sample project with the XMLtoSQL class library” on page 200. You can ignore this form at the moment.

Add a simple XSL stylesheet to test the optional XML format conversion

Since the demo servlet allows the XML transformation by applying an XSL stylesheet, we already have prepared a simple one, which will convert the generic XML into a custom XML format. To include the XSL file into your Web project, here is what you need to do:

1. While in the **J2EE Navigator** window, right-click the Web Content folder. Select **New** --> **Other** and then **XML** --> **XSL**. Click **Next**.
2. As the filename enter **ITSO_Customers.xsl**. Click **Finish**.
3. The empty XSL template file should open in a window. Replace the content with the XSL file from Example 8-12.

Example 8-12 ITSO_Customers.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0"
    xmlns:xalan="http://xml.apache.org/xslt">
<xsl:output method="xml" encoding="UTF-8" indent="yes" xalan:indent-amount="2"/>
<xsl:strip-space elements="*"/>
  <xsl:template match="/">
  <ITSO_Customers>
   <xsl:for-each select="/SQLResult/customer">
        <Customer_Address>
        <Customer_Num>
          <xsl:value-of select="customer_num"/>
        </Customer_Num>
        <First_Name>
          <xsl:value-of select="fname"/>
        </First_Name>
        <Last_Name>
          <xsl:value-of select="lname"/>
        </Last_Name>
        <Company>
          <xsl:value-of select="company"/>
        </Company>
        <Address_1>
          <xsl:value-of select="address1"/>
        </Address_1>
        <Address_2>
          <xsl:value-of select="address2"/>
        </Address_2>
        <City>
          <xsl:value-of select="city"/>
        </City>
        <State>
          <xsl:value-of select="state"/>
        </State>
        <Zipcode>
          <xsl:value-of select="zipcode"/>
        </Zipcode>
          <Phone>
          <xsl:value-of select="phone"/>
        </Phone>
          </Customer_Address>
     </xsl:for-each>
   </ITSO_Customers>
  </xsl:template>
</xsl:stylesheet>
```

Run the demo html/page servlet on an internal WSAD test server

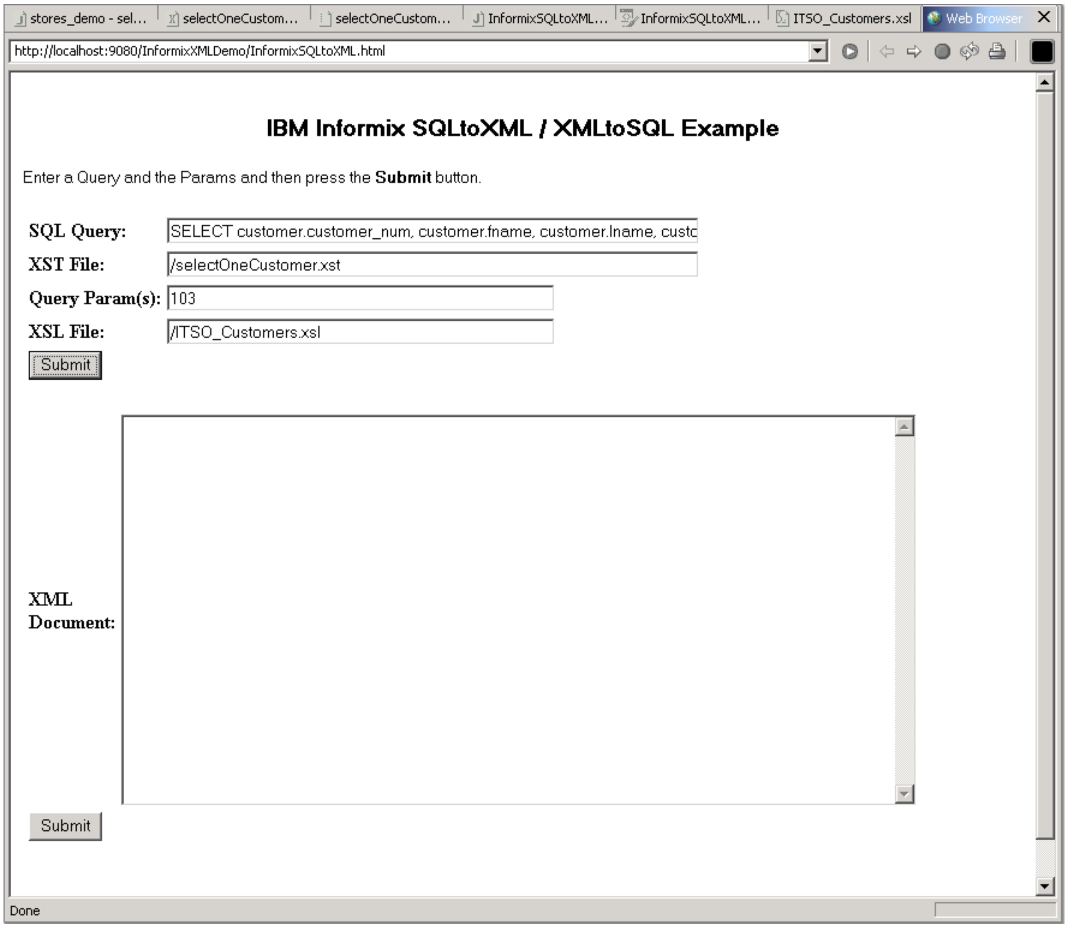
Now we're ready to publish and run the demo application. In order to do so, do the following:

1. Verify that your demo database is up and running
2. While being in the J2EE Navigator window, right-click the **InformixSQLtoXML.html** file and select **Run on Server**.
3. In the Server Selection Window, select from the pull down menu the appropriate WAS server test environment. A good choice for our demo would be the WebSphere V5.0 Test Environment and then click OK.
4. After a short while a Web Browser window should open and should look like Figure 8-6 on page 201.
5. As soon as you click the first Submit button (the second one is not working yet!) without modifying any of the input fields, SQLtoXML should execute the selectOneCustomer.xst file and apply the ITSO_Customers.xsl stylesheet. The final outcome of this process should look like Figure 8-7 on page 202.
6. Now you can try different values in the SQL Query input field, if you would like to try an ad-hoc query or supply a different XSL file in the XSL File input field.

8.3.4 Enhance the sample project with the XMLtoSQL class library

In the previous section we have explained how to utilize the SQLtoXML class library to generate XML from SQL. Now we will focus on the mapping from an existing XML document to relational tables in IDS.

For this purpose WSAD supports the XMLToSQL class library as part of the SQLToXML framework. Based on an existing XML document you can either use the interactive XMLToSQL wizard in WSAD to map a single XML file to an existing table or use the more flexible programmatic approach and utilize the class from within an application or servlet.



IBM Informix SQLtoXML / XMLtoSQL Example

Enter a Query and the Params and then press the **Submit** button.

SQL Query: SELECT customer.customer_num, customer.fname, customer.lname, custc

XST File: /selectOneCustomer.xst

Query Param(s): 103

XSL File: /ITSO_Customers.xsl

XML Document:

Figure 8-6 The Informix SQL to XML Demo Application

Let’s take a look at the basic requirements to make the XMLToSQL class work:

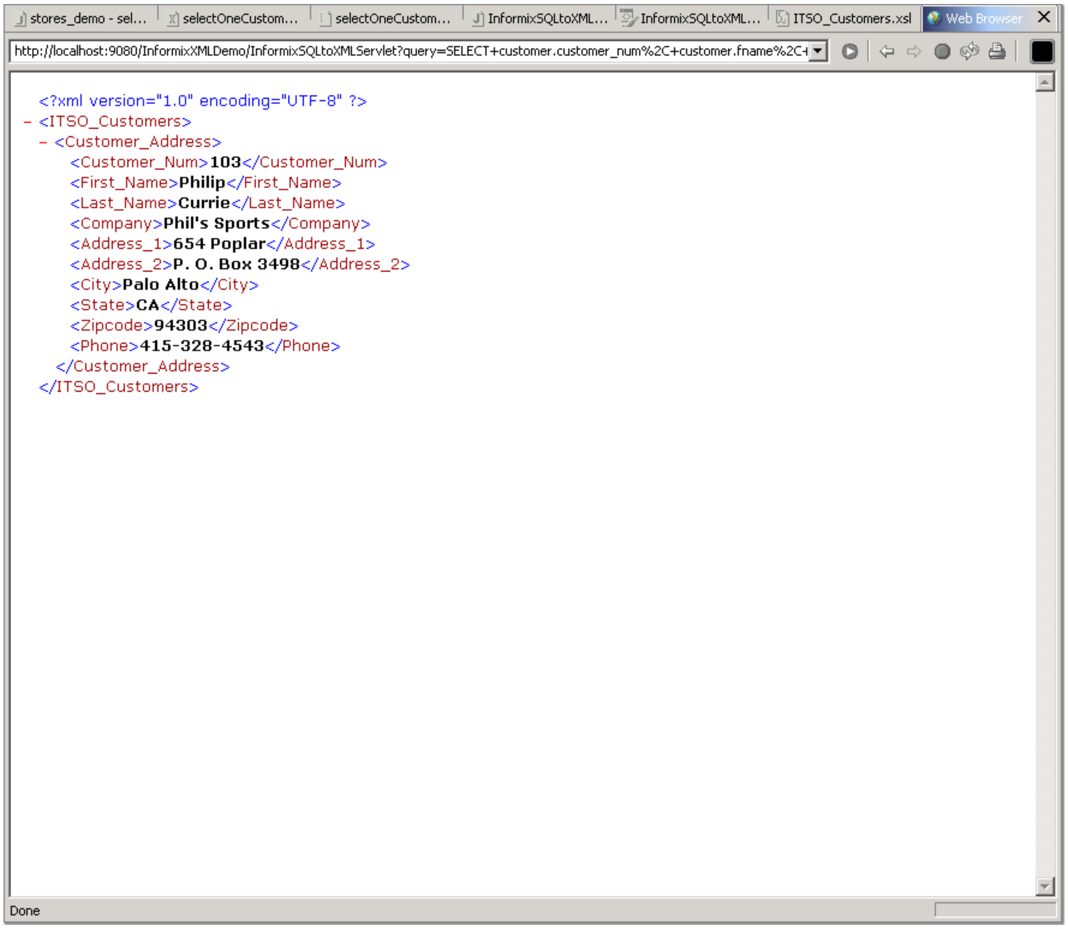
1. The root element of the XML document has to be <SQLResult>.
2. The next element(s) should represent the table names in the database
3. Each table element may have sub-elements which represent the columns in the table.
4. It is important that the names of the table and column elements exactly match the database table and column names
5. If your original XML input document doesn’t match the criteria above, you might have to write an XSL stylesheet to convert the original format into the required format.

XMLToSQL supports three different operation modes: INSERT, UPDATE, DELETE. In update mode the primary key cannot be updated.

Example 8-13 shows a valid XML document which can be used, for example, to insert a new row into the customer table into the stores_demo database.

Example 8-13 A valid XMLToSQL document

```
<?xml version="1.0" encoding="UTF-8"?>
<SQLResult>
    <customer>
        <customer_num>0</customer_num>
        <fname>Sabrina</fname>
        <lname>Koerner</lname>
        <company>Sabrinas Popcorn Parlour</company>
        <address1>6111 San Ignacio Ave.</address1>
        <address2>Appartment 301</address2>
        <city>San Jose</city>
        <state>CA</state>
        <zipcode>95119</zipcode>
        <phone>408-226-7676</phone>
    </customer>
</SQLResult>
```



http://localhost:9080/InformixXMLDemo/InformixSQLtoXMLServlet?query=SELECT+customer.customer_num%2C+customer.fname%2C+

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <ITSO_Customers>
  - <Customer_Address>
      <Customer_Num>103</Customer_Num>
      <First_Name>Philip</First_Name>
      <Last_Name>Currie</Last_Name>
      <Company>Phil's Sports</Company>
      <Address_1>654 Poplar</Address_1>
      <Address_2>P. O. Box 3498</Address_2>
      <City>Palo Alto</City>
      <State>CA</State>
      <Zipcode>94303</Zipcode>
      <Phone>415-328-4543</Phone>
    </Customer_Address>
  </ITSO_Customers>
```

Figure 8-7 Result of the Informix SQL to XML demo

OK, now we're modifying the Java demo servlet code to include support for the XMLToSQL class file.

Adding the XMLtoSQL support to the demo servlet source code

Switch to the J2EE Navigator by selecting **Window** --> **Open Perspective** --> **J2EE** and select the **J2EE Navigator** tab.

In the navigator window select the InformixSQLtoXMLServlet.java file within the Java Source folder.

In the editor window, replace the doPost() method with the following code (Example 8-14):

Example 8-14 The doPost() method for the demo servlet

```
   public void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException {

       String xmlInput = getParameter(req,"xmldoc");

       PrintWriter response = resp.getWriter();

       try
       {
          SQLProperties prop = new SQLProperties();

          prop.setJdbcDriver("com.informix.jdbc.IfxDriver");
          prop.setLoginId("itso");
          prop.setPassword("itso");

prop.setJdbcServer("jdbc:informix-sqli://neon.almaden.ibm.com:1533/stores_demo:INFORMIXSERV
ER=demo_on");
          prop.setAction(SQLProperties.INSERT);

          XMLToSQL xml2sql = new XMLToSQL(prop);

          xml2sql.execute(new ByteArrayInputStream(xmlInput.getBytes()));
       }
            catch (Exception e)
    {
      error(response, e);
    }
}
```

Now save the InformixSQLtoXMLServlet.java file into the Web project.

Finally re-run the demo HTML page as described in “Run the demo html/page servlet on an internal WSAD test server” on page 200.

You should see the HTML page displayed in the WSAD V5 built-in Web browser as shown in Figure 8-6 on page 201.

In the XML document field, enter a valid XML document which follows the guidelines for XMLToSQL documents. You could, for example, cut and paste the XML document from Example 8-13 on page 201.

As soon as you have entered the XML document correctly, click the **Submit** button.

To verify that the data has been correctly inserted into the customer table, you could either use the demo application input form and supply as a query parameter the customerid of the customer you just entered into the XML document field, or you could connect to the database directly by using, for example, dbaccess or ServerStudio.

To get familiar with different XMLToSQL behavior, you could modify the `prop.setAction(SQLProperties.INSERT)` line with the other supported operations (for example, `SQLProperties.UPDATE or SQLProperties.DELETE).`

If the source format of your XML document doesn’t match the required format for the XMLToSQL class library, you should include an XSL transformation before you call the `xml2sql.execute()` method.

You can do this by calling the already prepared `xmlTransform()` method in Example 8-10 on page 197.

Further information to SQLToXML and XMLToSQL

For your convenience we have included a detailed description of both Java classes in Appendix A, "SQLtoXML and XMLtoSQL Java class description" on page 311.

8.3.5 Additional dynamic XML mapping options in WSAD V5

In addition to the more database oriented mapping which we have described, WSAD V5 also supports the following Java oriented XML mappings:

- Generating XML/XSL Files from Java beans
- Generating Java beans from DTDs
- Generating Java beans from XML schema
- Generating an XML schema from a relational table
- Generating a relational table definition from an XML schema

The following sections describe briefly how to achieve the different tasks within WSAD V5.

Creating XSL and XML files from Java beans

You can use the Java Bean XML/XSL client wizard to take any Java bean (including EJBs) and convert the data in the bean into a DOM tree. The Java Bean XML/XSL client wizard also generates corresponding XSL stylesheets that work with the DOM tree. As well, a servlet is generated that will invoke the Xalan transformer to apply the XSL stylesheets to the DOM to produce an HTML form.

Follow these steps to create XML and XSL files from Java beans:

1. Switch to the XML perspective.
2. To launch the Java Bean XML/XSL client wizard, select **File** --> **New** --> **Other**. Then select **XML** --> **Java Bean XML/XSL Client** and click **Next**.
3. Click **Browse** to specify the location into which the generated XSL and XML files will be placed, as well as where all servlets will be mapped. The **Destination folder** must be the **Web Content** subfolder of a dynamic Web project. The **Java package** name specifies where all Java source code will be placed. A default package is assumed if you do not designate a package name.
4. The **Generate XML** model is the only model type you can use.
5. The **Files** field lists the type of XML and XSL files that will be generated from the Java bean. Select a file to see a description of it. Click **Next**.
6. Click **Browse** to locate the bean that will act as the model for your generated XML or XSL pages. Click **Introspect** to view the bean's methods. **Note: For the purposes of the wizard, any Java class that has a public constructor is considered to be a bean**.
7. Select the bean methods to be executed by the servlet. You can also use the **All** or **None** button to select or deselect the entire list. The wizard should make available the **public void methods() { }** that have primitive type parameters from your Java bean. The wizard also provides additional public methods inherited from any of the superclasses in the bean's hierarchy. Click **Next**.
8. Design the input form by specifying the page properties and the bean properties (fields) that the generated Web page or XML form will expose to the user for input. This page will show you the public properties of the bean, as well as any parameters that must be specified for the methods selected on the previous page. The scrolling panel on the right side of this page approximates the look and layout of the resulting page. You can also use

the **All** or **None** button to select or deselect the entire list of properties and method parameters. Click the **up** and **down** arrow buttons to reorder the columns in the input page. Click **Next**.

9. Design the results form by specifying the page properties and the bean properties (fields) that the generated XSL stylesheet will display as output to the user. The wizard should only make available the **public void getMethods() { }** from your Java bean. The available properties that you can specify for the result form include the following:

   Page: Page properties include Title, Background, Title Color, and Field Color.

   Fields: Field properties include ID, Label, Initial Value, Input Type, size, and max length. Note that the size property addresses the physical dimensions of the field, and the max length is a specification of the maximum number of characters or string length allowed.

   Click **Next**.

10. Provide a common prefix for the pages that are generated from the specified beans. Note that the list of generated pages and resources will reflect any changes to the prefix as you type in the **Prefix** field.
11. Click **Finish** to generate your Java code, XSL files, and XML schema file. Your Web project's web.xml file will also be updated.

Generating Java beans from a DTD

You can use the DTD editor to generate Java beans from a DTD file. Using these beans, you can quickly create or load an instance document that conforms to the DTD without coding directly to the DOM APIs.

To generate beans from a DTD file, follow these steps:

1. Create a project that is configured to work with Java source code. The beans you generate from your DTD must be contained in a project that is configured to work with Java source code. Projects such as a Java, Web, or Fragment project are all configured to work with Java source code. You do not have to store the DTD file in a project that is configured to work with Java source, however, we recommend it.
2. Switch to the XML perspective.
3. Open the DTD file you want to work with in the DTD editor.
4. Ensure that the DTD editor is in focus (otherwise, the appropriate menu will not be available). Click **DTD** --> **Generate Java Beans**.
5. In the Container field, specify your project. You can also specify a source folder in a project to contain your Java beans; that is, a folder that is on the Java source path. You can verify that a folder is on the Java source path by looking in the **Source** page of the Java Build Path for the project (Right-click the project and select **Properties** --> **Java Build Path** --> **Source**). Click Browse to select from a list of all valid projects and folders that exist in the workbench.
6. Type the name of the package that will contain the beans.
7. Select the name of the **Root element** for the beans. This can be any element in the DTD.
8. If desired, select the **Generate sample test program** check box.
9. Click **Finish**.
10. The beans appear in the Navigator view. Double-click them to edit them in the Java Editor.

The following beans will be generated:

1. A bean for each element in the DTD file. The name of the element will be used as the name of the bean.

2. A Factory bean for class construction. The name of the root element will be used to construct this class. For example, if the root element is *PurchaseOrder*, the factory class will be called *PurchaseOrderFactory*. You can use the RootElementFactory.java bean as a starting point for creating a new XML file
3. Optionally, a Sample main program - Sample.java. This bean shows you how to use the beans you have created. This bean is only generated if you selected the **Generate Sample Test Program** check box when you created the beans.

As well, the necessary JAR files are also added to the project so that you can build and run the beans.

Note: If you want to run the beans, switch to the Java perspective, and click **Run** --> **Run As** --> **Java Application**. Any output will be displayed in the **Console** view.

Generating Java beans from an XML schema

To allow developers to quickly build an XML application, the XML schema editor supports the generation of beans from an XML schema. Using these beans, you can quickly create an instance document or load an instance document that conforms to the XML schema without coding directly to the DOM APIs.

To generate beans from an XML schema, follow these steps:

1. Create a project that is configured to work with Java source code. The beans you generate from your XML schema must be contained in a project that is configured to work with Java source code. Projects such as a Java, Web, or Fragment project are all configured to work with Java source code. You do not have to store the XML schema file in a project that is configured to work with Java source, however, we recommend it.
2. If necessary, switch to the XML perspective.
3. Open the XML schema file in the XML schema editor.
4. Ensure that the XML schema editor is in focus (otherwise, the appropriate menu will not be available). On the main menu bar, click **XSD** --> **Generate Java Beans**.
5. In the **Container** field, specify your project. You can also specify a source folder in a project to contain your Java beans; that is, a folder that is on the Java source path. You can verify that a folder is on the Java source path by looking in the **Source** page of the Java Build Path for the project (Right-click the project and select **Properties** --> **Java Build Path** --> **Source**). Click **Browse** to select from a list of all valid projects and folders that exist in the workbench.
6. Type the name of the package.
7. Type the name of the **Root element** for the beans. This can be any complex type or simple type in the XML schema.
8. If desired, select the **Generate Sample Test Program** check box. Click **Finish**.
9. The beans appear in the Navigator view. Double-click them to edit them in the Java Editor. To run the beans, switch to the Java perspective and click **Run** --> **Java Application**. Any output will be displayed in the Console view.

The following files will have been generated:

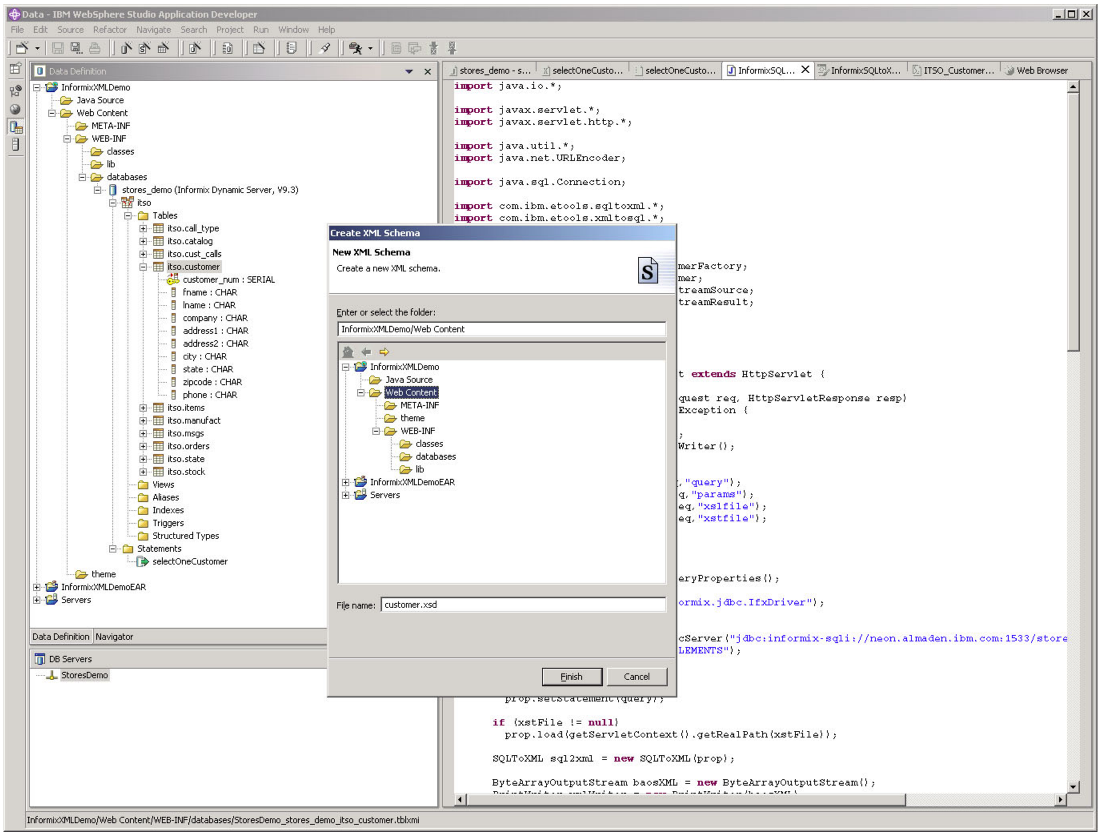
1. A bean for each construct (for example, a complex type, a global element) at the XML schema file level. The name of the construct will be used as the name of the bean.
2. A Factory class for class construction. The name of the root element will be used to construct this class. For example if the root element is *PurchaseOrder*, the factory class will be called *PurchaseOrderFactory*.

3. Optionally, a Sample main program.

As well, the necessary JAR files are also added to the project so that you can build and run the beans.

Generating an XML schema file from an Informix database table

You can generate an XML schema file from a relational table, and then further customize that file in the XML schema editor. See Figure 8-8.



Enter or select the folder: InformixXMLDemo/Web Content

File name: customer.xsd

```
import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

import java.util.*;
import java.net.URLEncoder;

import java.sql.Connection;

import com.ibm.etools.sqltoxml.*;
import com.ibm.etools.xmltosql.*;

merFactory;
mer;
treamSource;
treamResult;

t extends HttpServlet {

quest req, HttpServletResponse resp)
Exception {

;
Writer();

,"query");
q,"params");
eq,"xslfile");
eq,"xstfile");

eryProperties();

ormix.jdbc.IfxDriver");

cServer("jdbc:informix-sqli://neon.almaden.ibm.com:1533/store
LEMENTS");

  prop.setStatement(query);

if (xstFile != null)
  prop.load(getServletContext().getRealPath(xstFile));

SQLToXML sql2xml = new SQLToXML(prop);

ByteArrayOutputStream baosXML = new ByteArrayOutputStream();
```

Figure 8-8 The Generate XML SChema from table wizard

To generate a schema file from a relational table follow these steps:

1. Switch to the XML perspective.
2. Open the Data view (**Window** --> **Show View** --> **Other** --> **Data** --> **Data Definition**).
3. In the Data view, click the table that you want to work with.
4. From its pop-up menu, click **Generate XML Schema**.
5. Select a project or folder to contain the XML file and type a name for it. The name of the file must end in **.xsd**. Click **Finish**.
6. The XML schema appears in the Navigator view.

Generating an Informix database table definition from an XML schema

You can generate a relational table definition from an XML schema. To do so, follow these steps:

1. Switch to the XML perspective.
2. In the Navigator view, click the XML schema that you want to work with.
3. From its pop-up menu click **Generate** --> **DDL**.
4. Select the project or folder that will contain the relational table. In the File name field, type the name of the table, for example `MySchemaSQL.sql`. The name of your file must end with the extension **.sql**. Click **Finish**.
5. The file appears in the Navigator view. To open it, double-click it.

You should verify that it can be used to create an Informix database table.

Summary

This chapter has clearly shown that IDS is a very good foundation for XML related applications, and that WebSphere Application Developer provides a powerful framework to assist an Informix developer in XML development.

IDS 9, through its very flexible, extensible architecture, should support any kind of future requirements that an Informix based application might have.

In the next chapter we will focus on an even more advanced usage for XML: Web services in combination with IDS.

9

IDS, Web services, and WebSphere

XML based Web services provide a new way of heterogeneous application interaction by using standard protocols and the Internet infrastructure for communication.

In this chapter we discuss the use of IBM Informix Dynamic Server as a foundation for the implementation of Web services. We start with a brief introduction into Web services. Then we focus on IDS as a Web services provider, and later on, IDS as a Web services consumer.

We cover these topics:

- Introduction to Web services
- IDS as a Web Service provider
- IDS as a Web Service consumer

9.1 Introduction to Web services

Before we start the Web services introduction, we would like to point you to another redbook, which covers the topic of Web services in much greater detail: *WebSphere Version 5, Web Services Handbook*, SG24-6891. You can refer to this redbook if you have more specific questions regarding Web services.

In the current chapter, starting in 9.2, "Using IDS as a Web service provider" on page 214, we will focus more on how to integrate IBM IDS into an overall Web services framework.

Web services

A Web service is a set of related application functions that can be programmatically invoked over the Internet. Businesses can dynamically mix and match Web services to perform complex transactions with minimal programming. Web services allow buyers and sellers all over the world to discover each other, connect dynamically, and execute transactions in real time with minimal human interaction.

Web services are self-contained, self-describing modular applications that can be published, located, and invoked across the Web:

- **Web services are self-contained:** On the client side, a programming language with XML and HTTP client support is enough to get you started. On the server side, a Web server and servlet engine are required. The client and server can be implemented in different environments. It is possible to Web service enable an existing application without writing a single line of code.
- **Web services are self-describing:** The client and server need to recognize only the format and content of request and response messages. The definition of the message format travels with the message; no external metadata repositories or code generation tools are required.
- **Web services are modular:** Simple Web services can be aggregated to form more complex Web services either by using workflow techniques or by calling lower layer Web services from a Web service implementation.

Web services might be anything. Some examples, theatre review articles, weather reports, credit checks, stock quotations, travel advisories, or airline travel reservation processes. Each of these self-contained business services is an application that can easily integrate with other services, from the same or different companies, to create a complete business process. This interoperability allows businesses to dynamically publish, discover, and bind a range of Web services through the Internet.

Categories of Web services

Web services can be grouped into three categories:

- **Business information:** A business shares information with consumers or other businesses. In this case, the business is using Web services to expand its scope. Examples of business informational Web services are news streams, weather reports, or stock quotations.
- **Business integration:** A business provides transactional, "for fee" services to its customers. In this case, the business becomes part of a global network of value-added suppliers that can be used to conduct commerce. Examples of business integration Web services include bid and auction e-marketplaces, reservation systems, and credit checking.

- **Business process externalization:** A business differentiates itself from its competition through the creation of a global value chain. In this case, the business uses Web services to dynamically integrate its processes. Business process externalization Web services is exemplified by the associations between different companies to combine manufacturing, assembly, wholesale distribution, and retail sales of a particular product.

Service roles and interactions

A network component in a Web Services architecture can play one or more fundamental roles: service provider, service broker, and service client:

- **Service providers:** Create and deploy their Web services and can publish the availability of their services through a service registry, such as a UDDI Business Registry.
- **Service brokers:** Register and categorize published services and provide search services. For example, UDDI acts as a service broker for WSDL-described Web services.
- **Service clients:** Use broker services to discover a needed WSDL-described service on the UDDI Business Registry and then bind to and call the service provider.

Binding involves establishing all environmental prerequisites that are necessary to successfully complete the services. Examples of environmental prerequisites include security, transaction monitoring, and HTTP availability. The relationships between these roles are described in Figure 9-1.

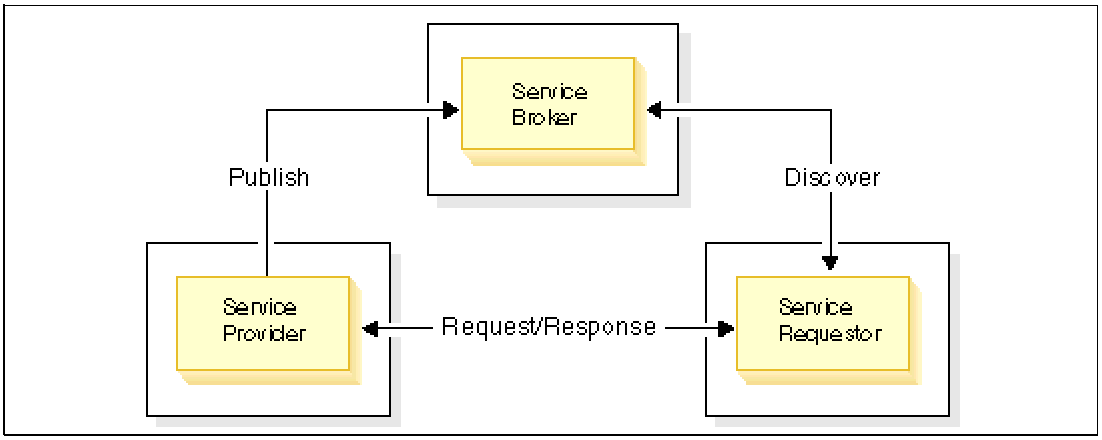


Figure 9-1 Service roles in a Web services scenario

9.1.1 Web service standards

One of the key attributes of Internet standards is that they focus on protocols and not on implementations. The Internet is composed of heterogeneous technologies that successfully interoperate through shared protocols. This prevents individual vendors from imposing a standard on the Internet. Open Source software development plays a crucial role in preserving the interoperability of vendor implementations of standards.

The following standards play key roles in Web services: Universal Description, Discovery and Integration (UDDI), Web Services Description Language (WSDL), Web Services Inspection Language (WSIL), and Simple Object Access Protocol (SOAP). The relationship between these standards is described in Figure 9-2.

The UDDI specification defines open, platform-independent standards that enable businesses to share information in a global business registry, discover services on the registry, and define how they interact over the Internet. For more information on UDDI, refer to `http://www.uddi.org`

WSIL is an XML-based open specification that defines a distributed service discovery method that supplies references to service descriptions at the service provider's point-of-offering, by specifying how to inspect a Web site for available Web services. A WSIL document defines the locations on a Web site where you can look for Web service descriptions. Since WSIL focuses on distributed service discovery, the WSIL specification complements UDDI by facilitating the discovery of services that are available on Web sites that may not be listed yet in a UDDI registry. A separate topic in this documentation discusses the relationship between UDDI and WSIL. For more information on WSIL, refer to:

`http://www.ibm.com/developerworks/webservices/library/ws-wsilspec.html`

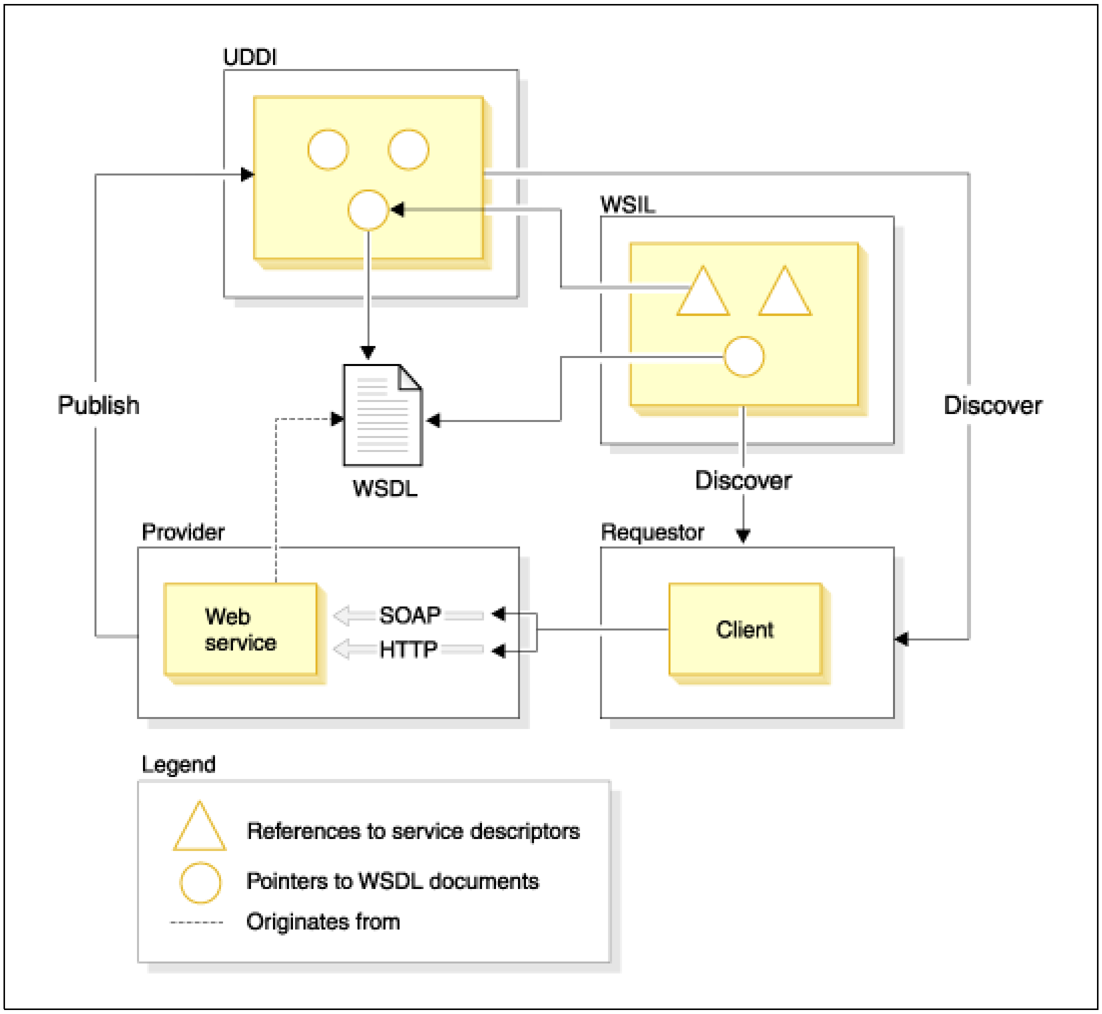


Figure 9-2 The roles of the different Web services standards

WSDL is an XML-based open specification that describes the interfaces to and instances of Web services on the network. It is extensible, so endpoints can be described regardless of the message formats or network protocols that are used to communicate. Businesses can publish the WSDL documents for their Web services to UDDI and discover the WSDL documents for other Web services in UDDI. WSDL is described as a separate topic in this documentation. For more information on WSDL, refer to `http://www.w3.org/TR/wsdl`

SOAP is an XML-based standard for messaging over HTTP and other Internet protocols. It is a lightweight protocol for the exchange of information in a decentralized, distributed environment. It is based on XML and consists of three parts:

- An envelope that defines a framework for describing what is in a message and how to process it.
- A set of encoding rules for expressing instances of application-defined data types.
- A convention for representing remote procedure calls and responses.

SOAP enables the binding and usage of discovered Web services by defining a message path for routing messages (Example 9-1). SOAP may be used to query UDDI for Web services. For more information on SOAP, refer to http://www.w3.org/TR/SOAP

Example 9-1 Sample SOAP message to call the getCustomer service method

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance">
   <SOAP-ENV:Body>
      <ns1:getCustomer encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="urn:itsoCustomerInfo"><customernum xsi:type="xsd:int">103</customernum>
</ns1:getCustomer>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A service provider hosts a Web service and makes it accessible using protocols such as HTTP GET, HTTP POST, and SOAP/HTTP. The Web service is described by a WSDL document that is stored on the provider's server or in a special repository. The WSDL document may be referenced by the UDDI business registry and WSIL documents. These contain pointers to the Web service's WSDL files.

For more information on Web services, refer to:

http://www.ibm.com/developerworks/webservices

Web services best practices

Although one could describe Web services simply as XML-protocol based remote function calls, you should avoid treating them as such for certain kinds of applications. Based on some real world projects which have been implemented at customer sites, most often without any IBM consultancy, we already have learned quite a few lessons. So, you need to watch out for the following issues if you're planning to implement Web services on top of your IBM Informix infrastructure:

- Do not use Web services between layers of an application or, for example, within a Java application server. The parsing of every Web service message is very costly and will slow down your application.
- Do not use Web services if you're not exposing external interfaces, for example, for interoperability or if you don't use an XML document based workflow.
- Use Web services on the edge of your application server to expose external APIs or if you need to execute remote calls through a firewall.
- If you have a need to execute function calls between Java application servers you might want to consider other protocols, for example, RMI/IIOP.

9.1.2 WSAD V5 tools for Web services development

WebSphere Studio provides the following tools to assist with Web services development:

- **Discover:** Browse the UDDI Business Registries or WSIL documents to locate existing Web services for integration. The Web becomes an extension of WebSphere Studio.

- **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, Informix IDS and DB2 User Defined Routines, and SQL statements (SELECT, INSERT, UPDATE).
- **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 Tomcat test environments using Server Tools.
- **Test:** Test Web services running locally or remotely in order to get instant feedback.
- **Develop:** Generate sample applications to assist you in creating your own Web service client application.
- **Publish:** Publish Web services to a UDDI v2 Business Registry, advertising your Web services so that other businesses and clients can access them.

These tools are accessed through the Web Services Client wizard, the Web Services DADX Group Configuration wizard, the Web Services wizard, Unit Test UDDI wizard, Java Beans for XML Schema wizard, and the IBM Web Services Explorer.

- Use the Web Services Client wizard to create the Java client to a deployed Web service and to test the Web service.
- Use the Web Services wizard to create, deploy, test, and publish Web services based on artifacts such as Java beans, URLs, enterprise beans, and, DADX files.
- The Web Services wizard supports the generation of the Java bean proxy and a sample application. Then publish your Web service to a UDDI Business Registry.
- The Unit Test UDDI wizard installs, configures, and removes a Private UDDI Registry.
- The Java Beans for XML Schema wizard enables you to generate Java beans from schema.
- The IBM Web Services Explorer assists you in discovering and publishing your Web service descriptions.

Web services tooling in WSAD V5 supports the following specifications:

- Web Services Definition Language (WSDL) Version 1.1
- Apache SOAP Version 2.3
- Universal Description, Discovery, and Integration Version 2.0
- Web Services Inspection Language (WSIL) Version 1.0

9.2 Using IDS as a Web service provider

Since we have now learned about the basics of Web services and the tools which are provided by WSAD V5 for Web services development, now let’s take a closer look at which kind of Web services are supported in combination with IDS:

9.2.1 IDS 7/9 Web services based on EJBs

This is a very straightforward process. In order to access an IDS based entity bean, you create a stateless session bean first and then use the Web services wizard in WSAD V5 to generate the necessary code for accessing the session bean.

Since these kind of Web services are more or less database independent due to the intermediate abstraction layer (session and entity beans) we didn't include any example in this redbook but instead refer you to another redbook which covers this topic in great detail: *Self-Study Guide, WebSphere Studio Application Developer and Web Services*, SG24-6407.

9.2.2 IDS 7/9 Web services based on Java beans

Using Java beans for IDS Web services is a very flexible and simple approach. The Java bean could contain either Informix JDBC calls to the database, IBM's data access bean code (a different abstraction layer to a pure JDBC application), calls to the SQLToXML and XMLToSQL class libraries, or even Java bean code which has been originally generated by the IBM Informix Object Translator (see "IBM Informix Object Translator (IDS 7/9)" on page 185 for details).

Since we have already developed a small servlet demo application in chapter 7 which is based upon the SQLToXML and XMLToSQL class libraries, let's take this Web project and enhance it with a Java bean based Web service.

Here are the steps we need to take to achieve this goal:

1. Create a new Java bean called InformixCustomerBean.
2. Use the WSAD V5 Web services wizard to create the WSDL files and generate a simple Java test client to test the new Web service.
3. Test the Web service with the generated Test client.

Create a new Java bean called InformixCustomerBean

Start WSAD V5 and open the workspace which you have used to develop the Web project in chapter 7.

Open the J2EE perspective by selecting **Window** --> **Open Perspective** --> **J2EE**

In the **J2EE Navigator** window, right-click the **Java Source** folder and then select **New** --> **Class**. In the **New Java Class** pop-up window just enter the name for the new Java bean: **InformixCustomerBean**. Click **Finish**. The empty bean should open automatically in an editor window.

The InformixCustomerBean.java code in Example 9-2 and Example 9-3 could be used as a template Java bean for any future SQLtoXML Web services development. For more details about that code, refer to the embedded comments. To get your first Web services demo application up and running, just follow the simple steps below.

Just insert the Java code (the import section) from Example 9-2 at the beginning of the file.

Example 9-2 The import section for the InformixCustomerBean.java file

```
import com.ibm.etools.sqltoxml.*;
import java.io.*;
import java.util.*;
import org.w3c.dom.*;

import javax.xml.transform.*;
import javax.xml.transform.dom.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
```

Now replace the *public class InformixCustomerBean { }* with the code in Example 9-3.

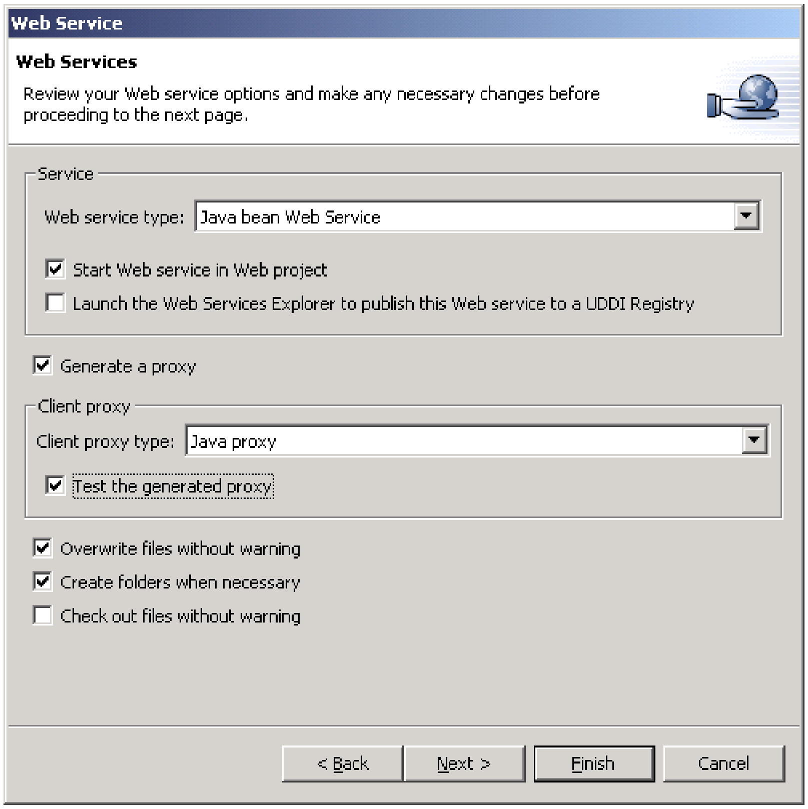
Example 9-3 The InformixCustomerBean class

```
public class InformixCustomerBean {

   public Element getOneCustomer(Integer customer_num) {

      Element returnResult = null;

      try {
         CharArrayWriter charWriter = new CharArrayWriter();
         PrintWriter writer = new PrintWriter(charWriter);
         QueryProperties props = new QueryProperties();

         // Locate the selectOneCustomer.xst file.  This file contains the
         // SQL statement.
         InputStream xstStream =
            this.getClass().getResourceAsStream("/selectOneCustomer.xst");

         props.load(xstStream);

         // Perform the query and generate an XML stream as output.
         // Note: this XML stream will be stored in the charWriter buffer
         // which we will switch to be an XML input stream below.
         SQLToXML sqltoxml = new SQLToXML(props);
         sqltoxml.setParameters(customer_num.toString());
         sqltoxml.setXMLWriter(writer);
         sqltoxml.execute();
         writer.flush();

         // Create a reader for the XML generated above for the
         // transformer code below.
         CharArrayReader charReader =
            new CharArrayReader(charWriter.toCharArray());

         // Now transform the XML generated by the query to the XML
         // format required for transmission to the Web services clients.

         // Create the transformation factory.
         TransformerFactory transFactory = TransformerFactory.newInstance();
         DocumentBuilderFactory buildFactory =
            DocumentBuilderFactory.newInstance();

         buildFactory.setNamespaceAware(true);

         // Create a stream for the XSL file.
         // This file contains the template of how the input XML stream
         // should be transformed into the output XML stream.
         InputStream xslStream =
            this.getClass().getResourceAsStream(
               "/ITSO_Customers.xsl");
         DocumentBuilder builder = buildFactory.newDocumentBuilder();
         Document xslDoc = builder.parse(xslStream);
         DOMSource xslSource = new DOMSource(xslDoc);

         xslSource.setSystemId("ITSO_Customers.xsl");

         // Create transformation code that uses the XSL document.
         Transformer transformer = transFactory.newTransformer(xslSource);

         // Create the xmlSource document from the XML stream create above.
         InputSource xmlInput = new InputSource(charReader);
         Document xmlDoc = builder.parse(xmlInput);
         DOMSource xmlSource = new DOMSource(xmlDoc);

         // The xmlResult object will contain the results of
         // the transformation.
         DOMResult xmlResult = new DOMResult();
         xmlSource.setSystemId("generatedXML.xml");

         // Perform the transformation.  The results are stored in xmlResult.
         transformer.transform(xmlSource, xmlResult);
         returnResult =
            ((Document) (xmlResult.getNode())).getDocumentElement();

      } catch (Throwable e) {
         System.out.println("Problem with sqltoxml:" + e.getMessage());
         e.printStackTrace();
      }
      // Send the response XML back to the Web services client.
      return returnResult;
   }
}
```

This Java bean now also references the same selectOneCustomer.xst and ITSO_Customers.xsl file from the demo servlet in Chapter 7. By applying an XSL file to the generic SQLToXML document, the Web service can be easily customized to the requirements of the clients which are utilizing the Web service. Save the modified code into the Web project.

Generate the WSDL files and the test client code

While still being in the J2EE perspective, right-click the **InformixCustomerBean.java** file in the **Java Source** directory of the **InformixXMLDemo** Web project. Select **New** --> **Other** --> **Web Services** --> **Web Service**.



Web Service

Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

Service

Web service type: Java bean Web Service

☑ Start Web service in Web project

□ Launch the Web Services Explorer to publish this Web service to a UDDI Registry

☑ Generate a proxy

Client proxy

Client proxy type: Java proxy

☑ Test the generated proxy

☑ Overwrite files without warning

☑ Create folders when necessary

□ Check out files without warning

Figure 9-3 The Web service pop-up window

In the **Web Services** pop-up window (Example 9-3) select **Java bean Web Service** as a Web service type from the pull down menu. Also select **Start Web service in Web project**, **Generate a proxy**, **Client proxy type: Java proxy**, **Test the generated proxy** and finally **Overwrite files without warning** and **Create folders when necessary**. Click **Next**.

In the **Web Service Deployment Settings** window keep all the pre-selected default values and then click **Next**.

In the **Web Service Java Bean Selection** window select the **InformixCustomerBean** bean and then click Next.

As with the two previous windows, you also shouldn't change anything in the **Web Service Java Bean Identity** window. Just click **Next**.

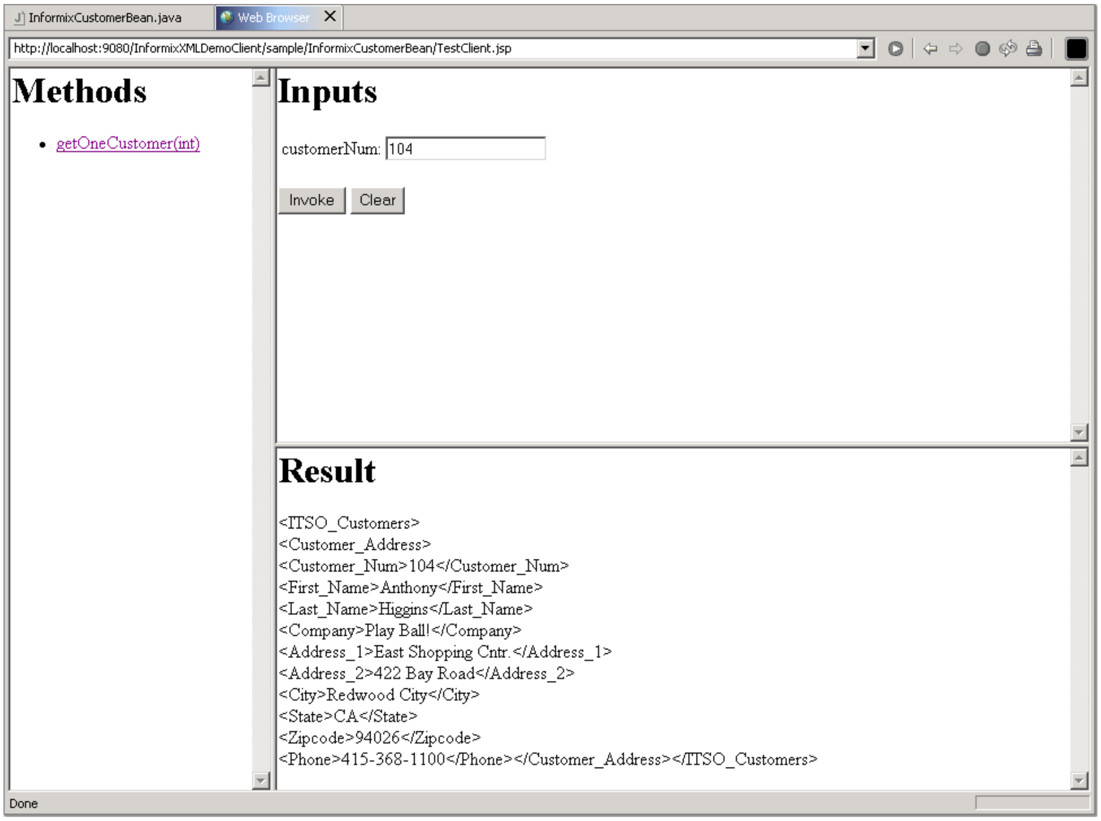
Leave everything as it is in the **Web Service Java Bean Methods** window. Click **Next**.

In the **Web Service Binding Proxy Generation** wizard select **Generate proxy**, if its not already pre-selected, and click **Next**.

In the **Methods** section of the **Web Service Test** wizard, de-select the methods *getEndPoint()* and *setEndPoint()*. Make sure that **Test the generated proxy** and **Run test on server** are selected. Then click **Next**.

On the final wizard window (**Web Service Publication**) you don't have to select anything, since we're not planning to publish our demo Web service to a UDDI register. Now click **Finish** to start the WSDL and test client generation process.

When the generation process completes, WSAD V5 will publish the newly created files, including the Java based test client, to the built-in WAS V5 test server and will then start an internal Web browser with the client JSP page displayed (Figure 9-4).


InformixCustomerBean.java
Web Browser
http://localhost:9080/InformixXMLDemoClient/sample/InformixCustomerBean/TestClient.jsp
Methods
getOneCustomer(int)
Inputs
customerNum: 104
Invoke
Clear
Result
Done

```
<ITSO_Customers>
<Customer_Address>
<Customer_Num>104</Customer_Num>
<First_Name>Anthony</First_Name>
<Last_Name>Higgins</Last_Name>
<Company>Play Ball!</Company>
<Address_1>East Shopping Cntr.</Address_1>
<Address_2>422 Bay Road</Address_2>
<City>Redwood City</City>
<State>CA</State>
<Zipcode>94026</Zipcode>
<Phone>415-368-1100</Phone></Customer_Address></ITSO_Customers>
```

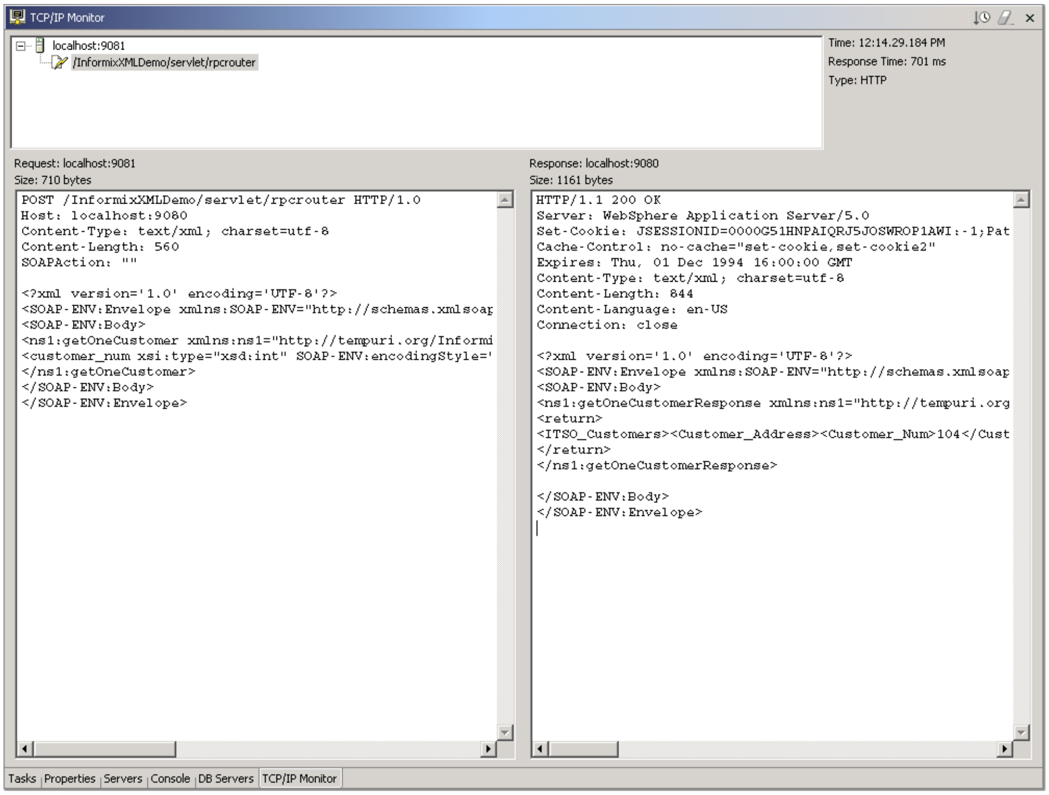
Figure 9-4 The generated Web service test client

In the Inputs window of the test client, enter a valid value into the customerNum input field and click **Invoke**.

The test client now calls the getOneCustomer method via the SOAP protocol and displays the result, converted via the ITSO_Customers.xsl stylesheet into the final format in the Result window.

Tip: Web services monitoring

Sometimes during the development cycle of a Web services application it can be very helpful to monitor what is going on behind the scenes — or more specifically, what is being transferred via the SOAP protocol.



Time: 12:14.29.184 PM
Response Time: 701 ms
Type: HTTP

Request: localhost:9081
Size: 710 bytes

```
POST /InformixXMLDemo/servlet/rpcrouter HTTP/1.0
Host: localhost:9080
Content-Type: text/xml; charset=utf-8
Content-Length: 560
SOAPAction: ""

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap
<SOAP-ENV:Body>
<ns1:getOneCustomer xmlns:ns1="http://tempuri.org/Informi
<customer_num xsi:type="xsd:int" SOAP-ENV:encodingStyle='
</ns1:getOneCustomer>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response: localhost:9080
Size: 1161 bytes

```
HTTP/1.1 200 OK
Server: WebSphere Application Server/5.0
Set-Cookie: JSESSIONID=0000G51HNPAIQRJ5JOSWROP1AWI:-1;Pat
Cache-Control: no-cache="set-cookie,set-cookie2"
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Content-Type: text/xml; charset=utf-8
Content-Length: 844
Content-Language: en-US
Connection: close

<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap
<SOAP-ENV:Body>
<ns1:getOneCustomerResponse xmlns:ns1="http://tempuri.org
<return>
<ITSO_Customers><Customer_Address><Customer_Num>104</Cust
</return>
</ns1:getOneCustomerResponse>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 9-5 Using the WSAD V5 built-in TCP/IP monitor

WSAD V5 has a built-in TCP/IP monitor which can be utilized to do this kind of monitoring.

How to activate the TCP/IP monitor? Just follow the next few steps to run the monitor:

1. Switch to the Server perspective by selecting **Window** --> **Open Perspective** -> **Server**.
2. In the Server Configuration window, right-click **Servers** and then select **New** --> **Server and Server Configuration**.
3. In the **Server name** input field, for example, enter TCP Monitor, and as a **Server type** select **TCP/IP Monitoring Server** from the list. Click **Next**.
4. In the Monitor Server Configuration keep the values at **Remote host** ‘localhost’ and **Remote port** ‘9080’. Click **Finish**.
5. Now you should see the newly created TCP Monitor entry in your **Server Configuration** window under **Servers**. The TCP/IP monitor should listen on port 9081 and forward any request to port 9080.
6. If you don’t see an active TCP/IP monitor window, select **Window** --> **Show View** --> (Other --> Server -->) **TCP/IP Monitor**.
7. Also make sure that your TCP/IP monitor has been started. Right-click the TCP Monitor entry in the Servers folder in the Server Configuration window and select **Control** --> **Start**.

In order to be able to monitor our test client application we only need to modify the InformixCustomerBeanProxy.java file to adjust the port number to the listen port of the TCP/IP monitor, instead of the original Web service port:

1. Switch to the **J2EE Navigator** window in the **J2EE perspective**. In the **InformixXMLDemoClient/Java Source** folder, open the **proxy.soap/InformixCustomerBeanProxy.java** file.
2. Replace the string: “http://localhost:9080/InformixXMLDemo/servlet/rpcrouter”
with “http://localhost:9081/InformixXMLDemo/servlet/rpcroute”
and save the file.

Now rerun the Test Client by doing the following steps:

1. Right-click the **TestClient.jsp** file in the **InformixXMLDemoClient/Web Content//sample/InformixCustomerBean** folder and select **Run on Server**.
2. Select your existing Test server environment and click **Finish**.
3. In the **Select a Server Client** window choose the **Web browser** only!
4. As soon as the Web Browser window opens, select the getOneCustomer method and enter a valid value in the **customerNum** input field (for example, 104).
5. Now click **Invoke**.
6. As soon as you got the results back you can switch to the TCP/IP monitor window (Figure 9-5).
7. In the upper window of the TCP/IP monitor, select the /**InformixXMlDemo/servlet/rpcrouter** entry.
8. In the lower two windows you should see now in the left window the SOAP message which has been sent to the Web service, and in the right window you should see the SOAP encoded reply.

Now you have seen how easy it is to develop a Web service based on Java beans. In the next section we will introduce an IDS based Web service which doesn’t even require you to write any Java code. You only need to know WSAD V5 and SQL!

9.2.3 DADX Web services and IDS 7/9

The document access definition extension (DADX) Web services had been originally developed with IBM DB2 and its XML Extender in mind. It allows you to easily wrap IBM DB2 XML Extender or regular SQL statements inside a Web service.

Fortunately, the non XML Extender related operations also work without any problems when using IBM Informix IDS 7 and 9. Let’s take a look at the supported DADX functions for IDS:

- Query
- Insert
- Update
- Delete
- Call Stored Procedures (limited support for IDS 7[1])

The runtime component of DADX Web services is called Web Services Object Runtime Framework (WORF). WORF uses the SOAP protocol and the DADX files and provides the following features:

- Resource based deployment and invocation
- Automatic service redeployment, at development time, when defining resource changes
- HTTP GET and POST bindings, in addition to SOAP
- Automatic WSDL and XSD generation, including support for UDDI Best Practices
- Automatic documentation and test page generation

[1] In IDS 7 you can only call stored procedures which do not return any results!

So how does WORF handle a Web service request in combination with IDS?

1. WORF receives an HTTP SOAP GET or POST service request.

   The URL of the request specifies a DADX or DTD file, and the requested action, which can be a DADX operation or a command, such as TEST, WSDL, or XSD. A DADX operation can also contain input parameters. WORF performs the following steps in response to a Web service request:

   – Loads the DADX file specified in the request.

   – Generates a response, based on the request.

     For operations:

     - Replaces parameters with requested values
     - Connects to IDS and runs any SQL statements, including UDR calls.
     - Formats the result into XML, converting types as necessary.

     For commands:

     - Generates necessary files, test pages, or other responses required.

   – Returns the response to the service requestor.

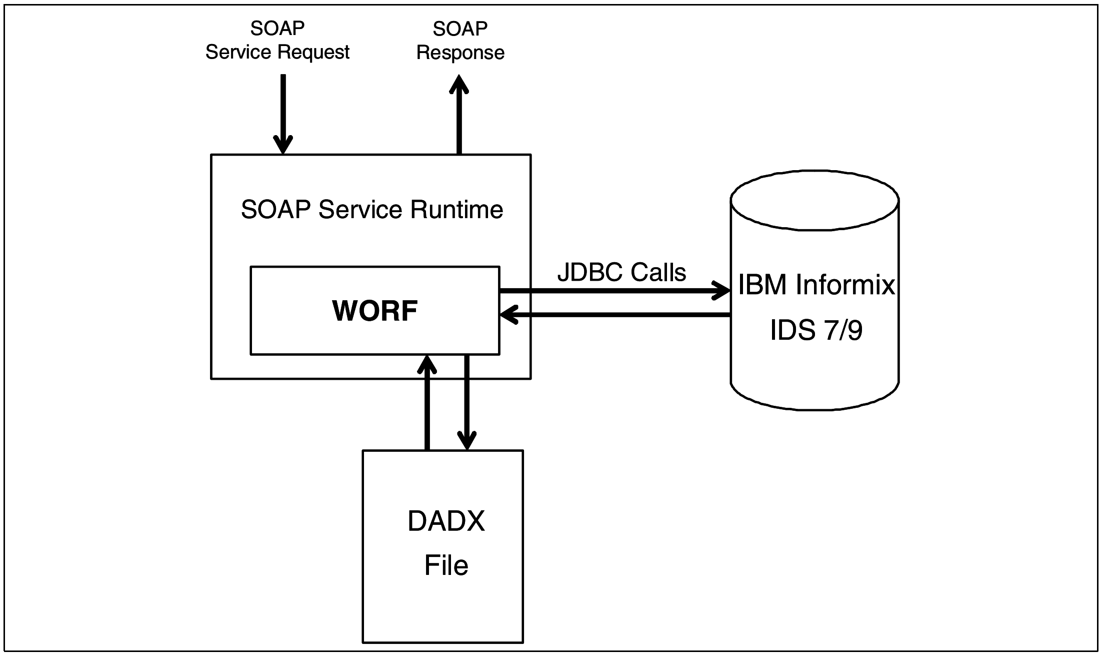


Figure 9-6 How WORF integrates IDS 7/9

Since it is so easy to implement an IDS based Web service with DADX Web services, in the next section we will take a look at how to develop such a service.

How to build a DADX Web service with IDS 7/9

In order to build a DADX Web service, we need to have some SQL statements on which the service should be based.

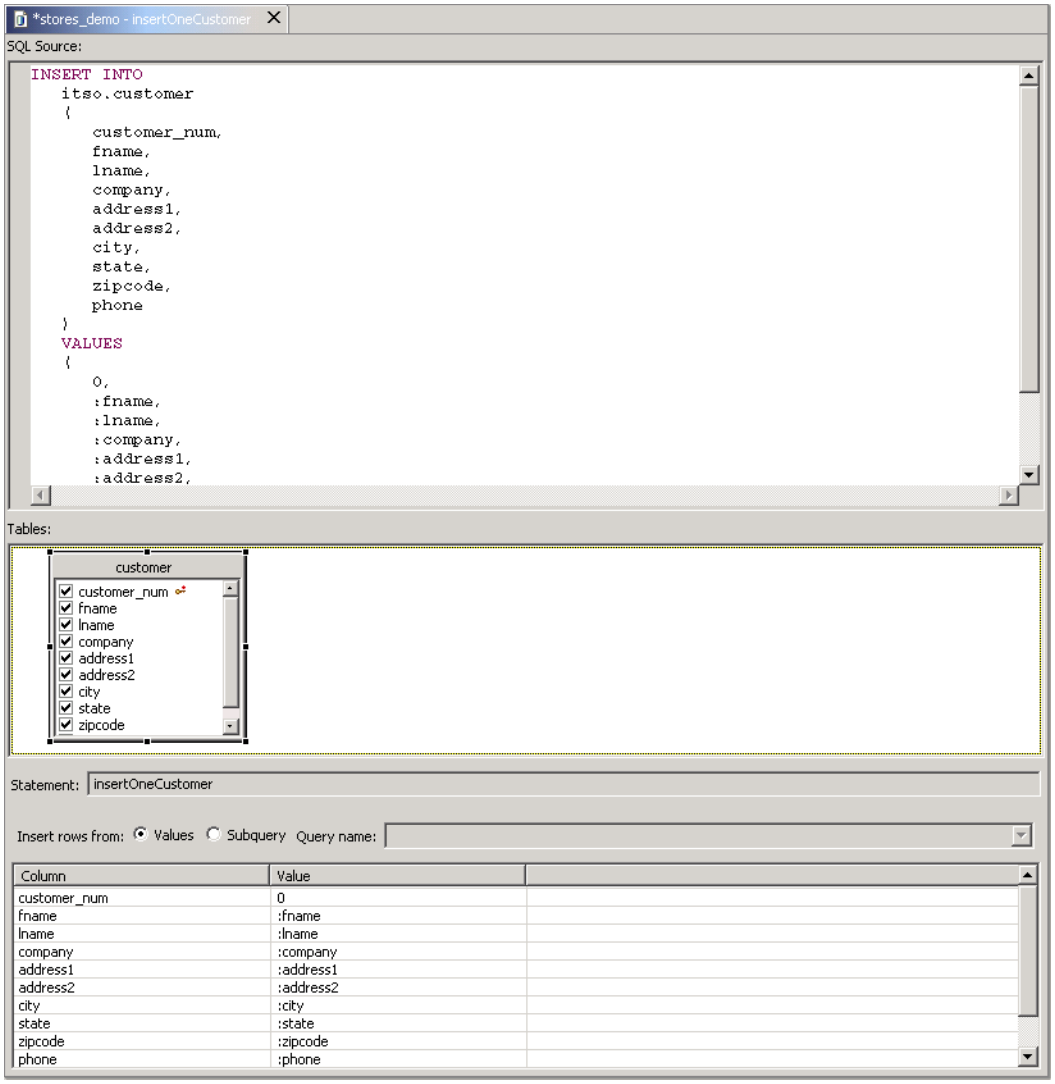
In our demo application we already defined one SELECT statement, named “selectOneCustomer” in “Define a SELECT statement by utilizing the Query Builder” on page 190 of Chapter 7. In addition to this already existing statement, we will define an INSERT statement to allow new additions to the customer table in the stores_demo database.

Add an addition SQL Statement for the demo DADX Web service

Open the WSAD V5 workspace which contains the demo Web project from Chapter 7 and 8. Switch to the **Data perspective** by selecting **Window** --> **Open Perspective** --> **Data**. In the **Data Definition** window open the **InformixXMLDemo/Web Content/databases/stores_demo** folder. Right-click the **Statements** folder and select **New** --> **Insert Statement**. Name the new statement **INSERTONECUSTOMER** and click **OK**.

In the interactive SQL builder window, right-click into the **Tables** window. Select **Add Table** and then from the tables selection menu, select the **itso.customer** table. Within the itso.customer table, select all attributes for the INSERT statement. In the window below the **Tables** window, we now have to define the host variables as placeholders for the later inserts which should be executed against the customer table.

To make it simple we'll name all host variables by using the column name with a colon (:) in front. To do this, click the **Value** column for each table attribute and enter the host variable name, for example, :fname for the fname attribute. Important: since the customer_num attribute is defined as a SERIAL data type in the database, we set the insert value to zero to automatically generate a new customer_num value during each insert! So eventually, the SQL builder window should look like Figure 9-7.



```
INSERT INTO
   itso.customer
   (
      customer_num,
      fname,
      lname,
      company,
      address1,
      address2,
      city,
      state,
      zipcode,
      phone
   )
   VALUES
   (
      0,
      :fname,
      :lname,
      :company,
      :address1,
      :address2,
```

☑ customer_num
☑ fname
☑ lname
☑ company
☑ address1
☑ address2
☑ city
☑ state
☑ zipcode

Statement: insertOneCustomer

Insert rows from: ● Values ○ Subquery Query name:

Column	Value
customer_num	0
fname	:fname
lname	:lname
company	:company
address1	:address1
address2	:address2
city	:city
state	:state
zipcode	:zipcode
phone	:phone

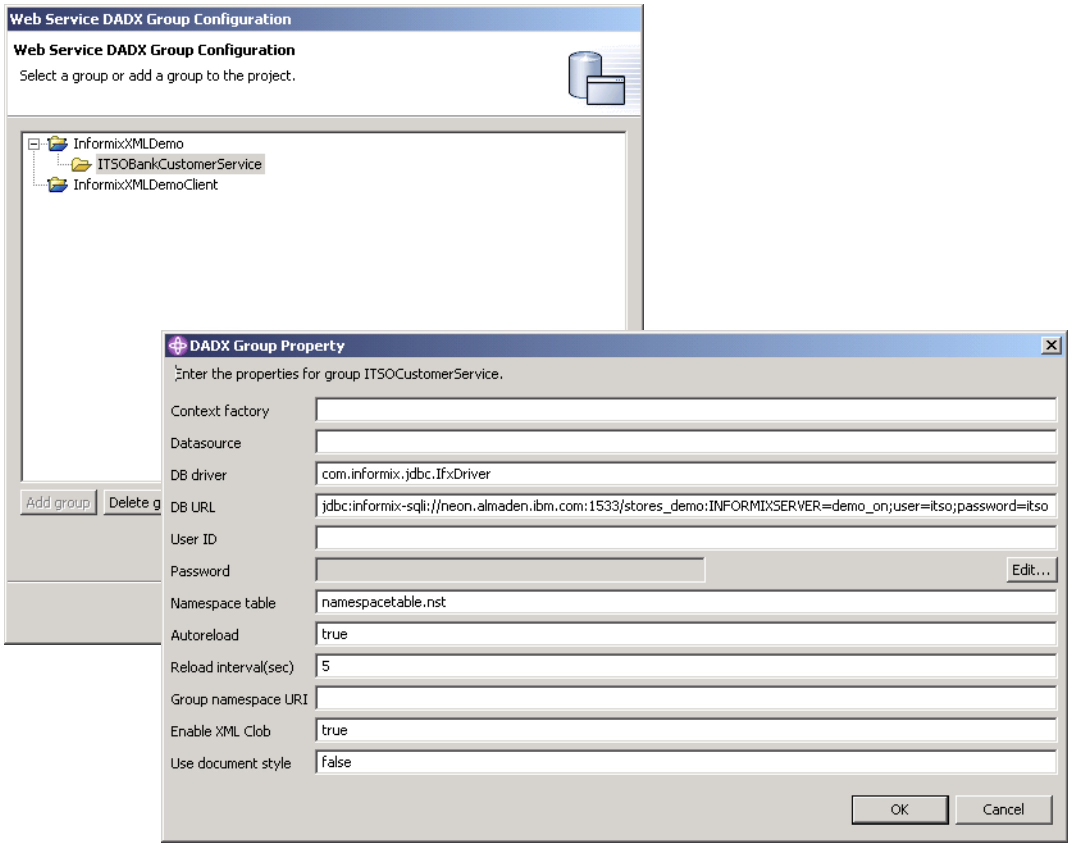
Figure 9-7 SQL Builder being used for the insertOneCustomer statement

As soon as you have defined the INSERT statement, save it into the demo Web project.

Create a DADX group and define its properties

In preparation for the to be generated DADX file, we need first to create a DADX group, which combines one or more SQL statement into one logical DADX Web service. So it could make sense, for example, to group all operations on the customer table into one group, while operations on the account table will be grouped into another DADX group.

Each DADX group also maintains its own database connection properties, so one could also use different DADX groups to connect to different databases or even different database servers (vendors).


Web Service DADX Group Configuration

Web Service DADX Group Configuration

Select a group or add a group to the project.

- InformixXMLDemo
  - ITSOBankCustomerService
- InformixXMLDemoClient

DADX Group Property

Enter the properties for group ITSOCustomerService.

Context factory:

Datasource:

DB driver: com.informix.jdbc.IfxDriver

DB URL: jdbc:informix-sqli://neon.almaden.ibm.com:1533/stores_demo:INFORMIXSERVER=demo_on;user=itso;password=itso

User ID:

Password:

Namespace table: namespacetable.nst

Autoreload: true

Reload interval(sec): 5

Group namespace URI:

Enable XML Clob: true

Use document style: false

Figure 9-8 Create a new DADX Group and set the DADX group properties for IDS

To create a new DADX group:

1. Open the **J2EE Perspective** by selecting **Window** --> **Open Perspective** --> **J2EE**. Then select the J2EE Navigator window.
2. Select **File** --> **New** --> **Other** --> **Web Services** --> **Web Service DADX Group Configuration**.
3. In the next window, select the **InformixXMLDemo** folder and then click **Add group**.
4. For the group name, enter **ITSOCustomerService**. Click **OK**.
5. While still being in the same window, now select the **InformixXMLDemo/ITSOCustomerService** folder and then click **Group properties**.
6. In the **DADX Group Properties** pop-up window, fill in the following information:

   DB driver: `com.informix.jdbc.IfxDriver`

DB URL:
`jdbc:informix-sqli://neon.almaden.ibm.com:1533/stores_demo:INFORMIXSERVER=demo_on;user=itso;password=itso`

7. Leave the other fields as-is. Your entries should look like Figure 9-8. Click **OK**.
8. In the **DADX Group Configuration** window, click **Finish**.

Generate the DADX file

Now we can generate the DADX file for the two SQL Statements. To do this:

1. Select **File** --> **New** --> **Other** --> **Web Services** --> **DADX File**. Click **Next**.
2. In the **Select SQL Statements** window, open the **InformixXMlDemo/Web Content/WEB-INF/databases/stores_demo/Statements** folder
3. Since we would like to select both SQL statements (insertOneCustomer, selectOneCustomer) we need to do the following: Click the **insertOneCustomer** statement first and then control-click the **selectOneCustomer** too. Now both statements should be selected (highlighted). Click **Next**.
4. Just click **Next** in the **Select DAD Files** window since DAD files are not yet supported with IBM Informix IDS.
5. In the DADX Generation window enter as the File name: **ITSOCustomerService.dadx**. Don't change the other values. Click **Finish**.

The generated ITSOCustomerService.dadx file should look the one in Example 9-4. Notice the XML compliant format and the specific DADX keywords. You'll find a complete description of the DADX file format in Appendix B, "DADX file format" on page 327.

Example 9-4 ITSOCustomerService.dadx file

```
<?xml version="1.0" encoding="UTF-8"?>
<dadx:DADX xmlns:dadx="http://schemas.ibm.com/db2/dxx/dadx"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xsi:schemaLocation="http://schemas.ibm.com/db2/dxx/dadx dadx.xsd">
    <dadx:operation name="selectOneCustomer">
        <dadx:documentation xmlns="http://www.w3.org/1999/xhtml">

        </dadx:documentation>
        <dadx:query>
            <dadx:SQL_query>
<![CDATA[
                SELECT itso.customer.customer_num, itso.customer.fname,
itso.customer.lname, itso.customer.company, itso.customer.address1, itso.customer.address2,
itso.customer.city, itso.customer.state, itso.customer.zipcode, itso.customer.phone FROM
itso.customer WHERE itso.customer.customer_num = :customernum
]]>
            </dadx:SQL_query>
            <dadx:parameter name="customernum" type="xsd:int"/>
        </dadx:query>
    </dadx:operation>
    <dadx:operation name="insertOneCustomer">
        <dadx:documentation xmlns="http://www.w3.org/1999/xhtml">

        </dadx:documentation>
        <dadx:update>
            <dadx:SQL_update>
<![CDATA[
                INSERT INTO itso.customer ( customer_num, fname, lname, company, address1,
address2, city, state, zipcode, phone ) VALUES ( 0, :fname, :lname, :company, :address1,
:address2, :city, :state, :zipcode, :phone )
]]>
            </dadx:SQL_update>
            <dadx:parameter name="fname" type="xsd:string"/>
            <dadx:parameter name="lname" type="xsd:string"/>
            <dadx:parameter name="company" type="xsd:string"/>
            <dadx:parameter name="address1" type="xsd:string"/>
            <dadx:parameter name="address2" type="xsd:string"/>
            <dadx:parameter name="city" type="xsd:string"/>
            <dadx:parameter name="state" type="xsd:string"/>
            <dadx:parameter name="zipcode" type="xsd:string"/>
            <dadx:parameter name="phone" type="xsd:string"/>
        </dadx:update>
    </dadx:operation>
</dadx:DADX>
```

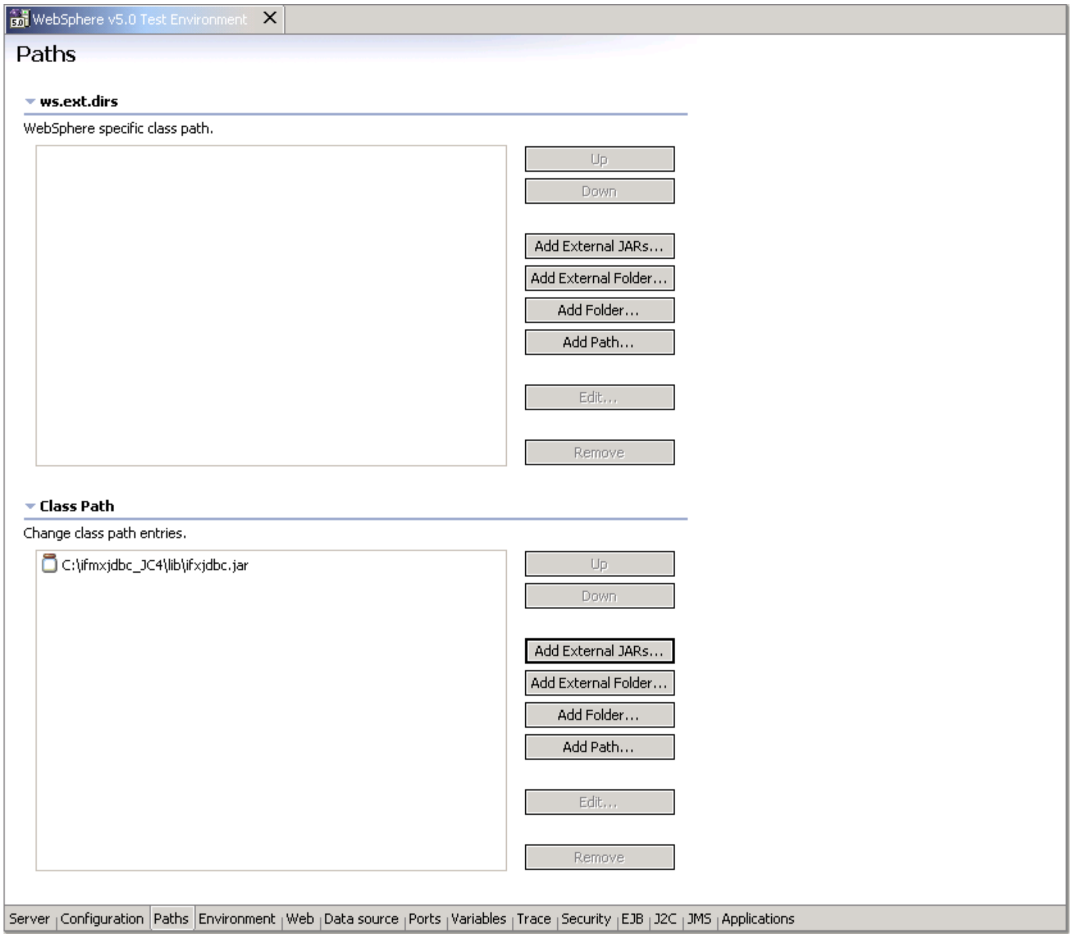
Since the interactive query builder in WSAD V5 only supports SELECT, INSERT, UPDATE and DELETE statements you might have to edit the generated DADX file manually if you want to add support for IDS user defined routines (UDR). For more information about UDR support in DADX files, refer to “DADX support for user defined routines / stored procedures” on page 229.

Create a DADX Web service based on the generated DADX file

Now let’s generate the necessary files for a DADX Web service based on the DADX file we generated in the previous section.

First we need to prepare the WSAD V5 built-in WebSphere V5 Test Environment for Informix IDS database access in combination with the DADX Web service:

1. Open the Server perspective by selecting **Window** --> **Open Perspective** --> **Server**.
2. In the **Server Configuration** window select your **WebSphere V5 Test Environment** with a double-click.



Paths

ws.ext.dirs

WebSphere specific class path.

Class Path

Change class path entries.

C:\ifmxjdbc_JC4\lib\ifxjdbc.jar

Figure 9-9 WebSphere V5 Test Environment configuration window

3. In the WebSphere V5 configuration window, select the **Paths** tab.
4. Now add the Informix JDBC driver to the Class Path entries by clicking **Add External JARs**. In the file browser select the correct ifxjdbc.jar file and click **Open** (Figure 9-9).
5. Save the server configuration into the demo Web project and close the server configuration window (**important!**).

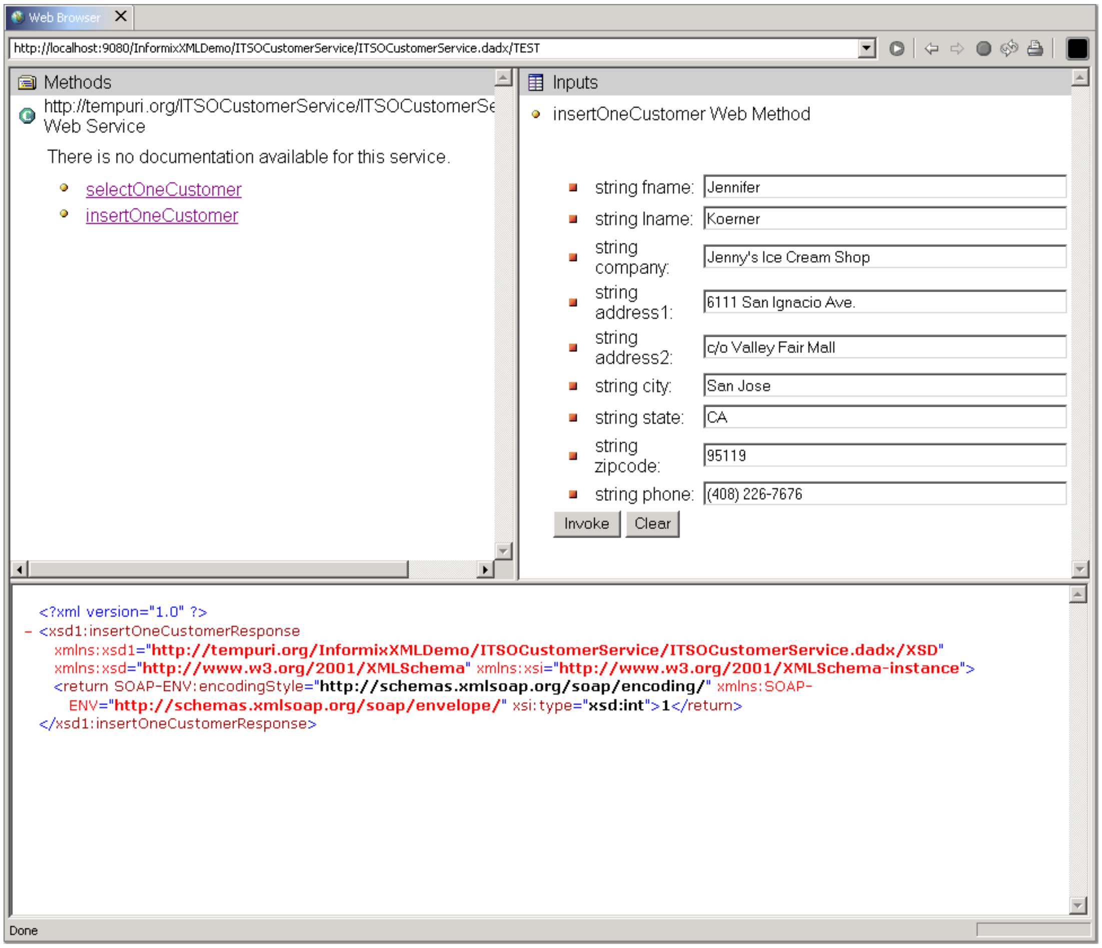
Now we build the Web service itself:

1. Open the **J2EE perspective** and click the file **ITSOCustomerService.dadx**
2. Select **File** --> **New** --> **Other** --> **Web Services** --> **Web Service**. Click **Next**.
3. In the **Web Services** window select as the **Web service type**: **DADX Web Servic**e. In addition, check the **Start Web service in Web project** option and the options, **Overwrite files without warning** and **Create folders when necessary**. Click **Next**.
4. In the **Web Service Deployment Settings** leave the default values and as the **Web project** choose **InformixXMLDemo**. Click **Next**.
5. The **DADX file** in the **Web Service DADX File Selection** window should be already set to:

   ```
   /InformixXMLDemo/Java Source/groups/ITSOCustomerService/ITSOCustomerService.dadx
   ```

   If not, enter this file name or use the **Browse** button to navigate to that file. Click **Next**.

6. In the **Web Service DADX Group properties** window verify that the **DB driver** and the **DB URL** input fields are set correctly (see Section , “Create a DADX group and define its properties” on page 224). Click **Next**.
7. In the **Web Service Binding Proxy Generation** window, just click **Finish**, since we don’t want to generate a Java proxy for testing purposes. We’re going to use the built-in test client of the WORF framework.


Web Browser
http://localhost:9080/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/TEST
Methods
Inputs
Invoke
Clear
Done

http://tempuri.org/ITSOCustomerService/ITSOCustomerSe
Web Service

There is no documentation available for this service.

- selectOneCustomer
- insertOneCustomer

insertOneCustomer Web Method

- string fname: Jennifer
- string lname: Koerner
- string company: Jenny's Ice Cream Shop
- string address1: 6111 San Ignacio Ave.
- string address2: c/o Valley Fair Mall
- string city: San Jose
- string state: CA
- string zipcode: 95119
- string phone: (408) 226-7676

```
  <?xml version="1.0" ?>
- <xsd1:insertOneCustomerResponse
    xmlns:xsd1="http://tempuri.org/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/XSD"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <return SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
      ENV="http://schemas.xmlsoap.org/soap/envelope/" xsi:type="xsd:int">1</return>
  </xsd1:insertOneCustomerResponse>
```

Figure 9-10 Test client for the DADX Web service: test of the insertOneCustomer method

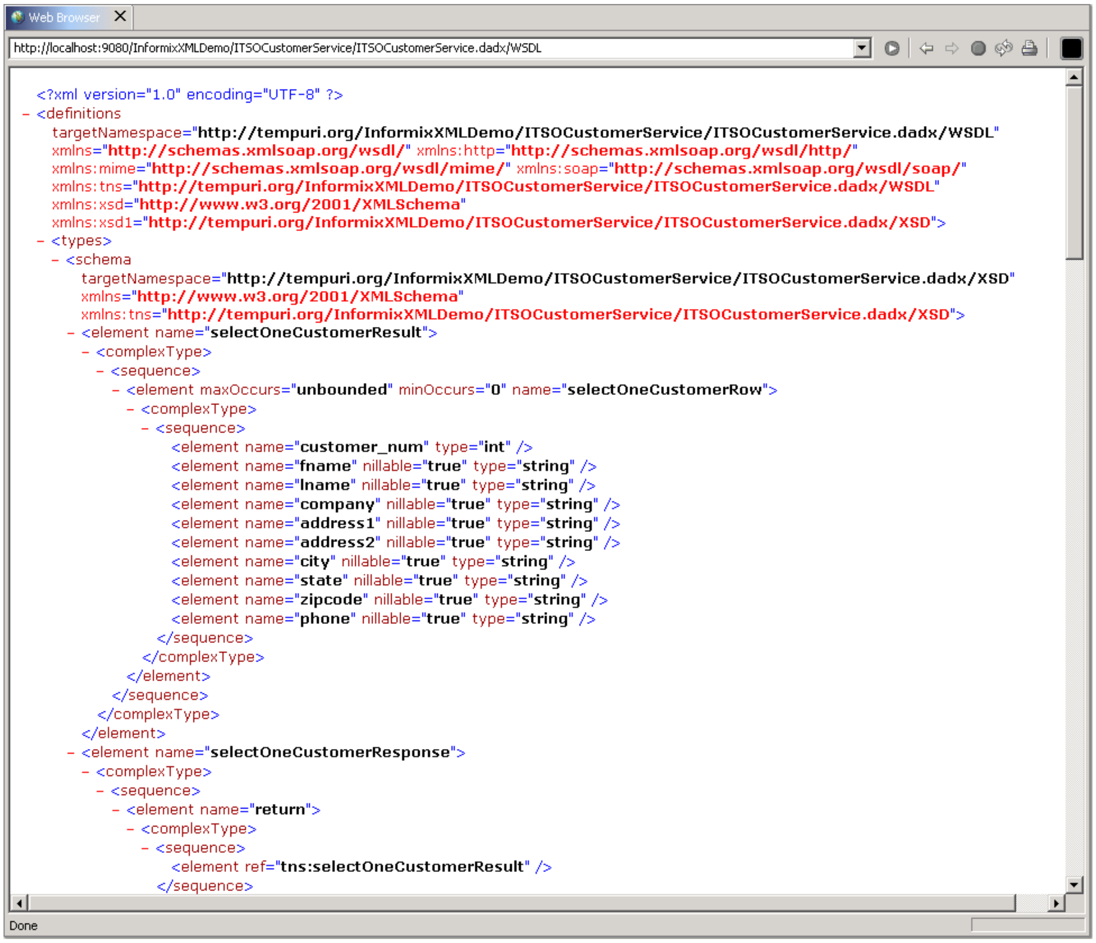
Let’s test the newly created Web service with the built-in Web browser:

1. Open the Server perspective again.
2. Click the Open Web Browser icon located on the toolbar of Server Perspective window.
3. In the Web browser address field, enter the following URL:

```
http://localhost:9080/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/TEST
```

4. This URL will activate the WORF built-in test client and allows the access to the predefined DADX Web service methods (see Figure 9-10 for the results of an insertOneCustomer method test).
5. Alternatively you could also generate a WSDL file (Figure 9-11) for your DADX Web service by opening the following URL in the Web browser:

```
http://localhost:9080/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/WSDL
```



Web Browser
http://localhost:9080/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/WSDL
Done

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <definitions
    targetNamespace="http://tempuri.org/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/WSDL"
    xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://tempuri.org/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/WSDL"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd1="http://tempuri.org/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/XSD">
  - <types>
    - <schema
        targetNamespace="http://tempuri.org/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/XSD"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://tempuri.org/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/XSD">
      - <element name="selectOneCustomerResult">
        - <complexType>
          - <sequence>
            - <element maxOccurs="unbounded" minOccurs="0" name="selectOneCustomerRow">
              - <complexType>
                - <sequence>
                    <element name="customer_num" type="int" />
                    <element name="fname" nillable="true" type="string" />
                    <element name="lname" nillable="true" type="string" />
                    <element name="company" nillable="true" type="string" />
                    <element name="address1" nillable="true" type="string" />
                    <element name="address2" nillable="true" type="string" />
                    <element name="city" nillable="true" type="string" />
                    <element name="state" nillable="true" type="string" />
                    <element name="zipcode" nillable="true" type="string" />
                    <element name="phone" nillable="true" type="string" />
                  </sequence>
                </complexType>
              </element>
            </sequence>
          </complexType>
        </element>
      - <element name="selectOneCustomerResponse">
        - <complexType>
          - <sequence>
            - <element name="return">
              - <complexType>
                - <sequence>
                    <element ref="tns:selectOneCustomerResult" />
                  </sequence>
```

Figure 9-11 WSDL generated by the DADX Web service

DADX support for user defined routines / stored procedures

In addition to standard SQL statements like SELECT, INSERT, DELETE and UPDATE, the WORF framework also supports the execution of user defined routines or stored procedures in IDS. In order to do this, the framework utilizes (internally) the JDBC CallableStatement class which is a very portable way of calling stored procedures and functions in database servers.

Since this feature is unfortunately not supported through the interactive SQL builder in WSAD, we need to either create a new DADX file or modify an existing one.

Before we go ahead with an example, let's take a look a the DADX file syntax for stored procedures/functions based on the XML schema for DADX files (see Example 9-5).

Example 9-5 DADX call operation (XML schema definition)

```
<element name="call">

    <annotation>
      <documentation>
        Calls a stored procedure.
        The call statement contains in, out, and in/out parameters using host variable
syntax.
        The parameters are defined by a list of parameter elements that are uniquely named
        within the operation.
      </documentation>
    </annotation>

    <complexType>
      <sequence>
        <element name="SQL_call" type="string"/>
        <element ref="dadx:parameter" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="dadx:result_set" minOccurs="0" maxOccurs="unbounded"/>
      </sequence>
    </complexType>

    <unique name="callParameterNames">
      <selector xpath="dadx:parameter"/>
      <field xpath="@name"/>
    </unique>

    <unique name="callResultSetNames">
      <selector xpath="dadx:result_set"/>
      <field xpath="@name"/>
    </unique>

</element>
```

As mentioned earlier, the WORF framework utilizes the JDBC java.sql.CallableStatement interface for the execution of IDS user defined routines. Therefore the syntax for calling routines in IDS this way should follow the JDBC guidelines. For a simple example in DADX syntax how to call a user defined routine which doesn't return any results, take a look at Example 9-6.

Example 9-6 Simple UDR call in DADX syntax

```
<dadx:operation name="createOneCustomerSimple">
   <dadx:documentation xmlns="http://www.w3.org/1999/xhtml">
   </dadx:documentation>
   <dadx:call>
      <dadx:SQL_call>
<![CDATA[
            { call create_customer_simple (:fname, :lname, :company, :address1, :address2,
:city, :zipcode, :state, :phone)}
]]>
      </dadx:SQL_call>
      <dadx:parameter name="fname" type="xsd:string" kind="in"/>
      <dadx:parameter name="lname" type="xsd:string" kind="in"/>
      <dadx:parameter name="company" type="xsd:string" kind="in"/>
      <dadx:parameter name="address1" type="xsd:string" kind="in"/>
      <dadx:parameter name="address2" type="xsd:string" kind="in"/>
      <dadx:parameter name="city" type="xsd:string" kind="in"/>
      <dadx:parameter name="zipcode" type="xsd:string" kind="in"/>
      <dadx:parameter name="state" type="xsd:string" kind="in"/>
      <dadx:parameter name="phone" type="xsd:string" kind="in"/>
   </dadx:call>
</dadx:operation>
```

If you have the need to return results back to the DADX Web service consumer you might have different options:

- Starting with IDS 9 you can utilize multiple out parameters in the UDR parameter list. To use these out parameters in combination with DADX you need to declare them as in/out parameters and the Web service caller might have to supply dummy values (for example, zero for integer types) to make it work. This behavior seems to be IDS specific and doesn't apply to other databases. The UDR create_customer_out in Example 9-7 shows a simple SPL UDR which uses one out parameter (newcustomernum). Example 9-8 shows the correct DADX syntax for calling such a UDR. Notice the in/out option for the newcustomernum parameter.

> **Tip:** This restriction seems to be specific to the DADX/IDS combination, since a similar restriction had been already removed in the IBM Informix JDBC 2.21.JC4 driver (which had been used for the examples in this redbook) and is no longer valid. Callers need to use registerOUTparameter() only and they do not need to use setXXX() method on OUT parameters. A future version of DADX will very likely address this change in the Informix JDBC driver.

Example 9-7 IDS 9 UDR with an out parameter (in SPL)

```
create procedure create_customer_out (fname lvarchar, lname lvarchar,
                company lvarchar, address1 lvarchar, address2 lvarchar,
                city lvarchar, zipcode lvarchar, state lvarchar,
                phone lvarchar, OUT customernum int)

   define new_customernum int;
   insert into customer values (0, fname, lname, company, address1,
                                      address2, city, zipcode, state, phone);
   let new_customernum = dbinfo('sqlca.sqlerrd1');
   let customernum = new_customernum;

end procedure;
```

Example 9-8 DADX syntax fragment for the IDS UDR from Example 9-7

```
<dadx:operation name="createOneCustomer">
   <dadx:documentation xmlns="http://www.w3.org/1999/xhtml">
   </dadx:documentation>
   <dadx:call>
      <dadx:SQL_call>
<![CDATA[
            { call create_customer_out (:fname, :lname, :company, :address1, :address2,
:city, :zipcode, :state, :phone, :newcustomernum)}
]]>
      </dadx:SQL_call>
      <dadx:parameter name="fname" type="xsd:string" kind="in"/>
      <dadx:parameter name="lname" type="xsd:string" kind="in"/>
      <dadx:parameter name="company" type="xsd:string" kind="in"/>
      <dadx:parameter name="userid" type="xsd:string" kind="in"/>
      <dadx:parameter name="address1" type="xsd:string" kind="in"/>
      <dadx:parameter name="address2" type="xsd:string" kind="in"/>
      <dadx:parameter name="city" type="xsd:string" kind="in"/>
      <dadx:parameter name="zipcode" type="xsd:string" kind="in"/>
      <dadx:parameter name="state" type="xsd:string" kind="in"/>
      <dadx:parameter name="phone" type="xsd:string" kind="in"/>
      <dadx:parameter name="newcustomernum" type="xsd:int" kind="in/out"/>
   </dadx:call>
</dadx:operation>
```

- You could simply return a result for a UDR or even complete results sets. See the following important tip regarding the support in IDS for that feature.

Tip: IDS 9.40 supports a new feature which allows that the columns of a result set for a user defined routine can have display labels. The WORF framework requires the usage of those labels in IDS or one couldn't use UDRs with result sets. Unfortunately there seem to be an issue in the release 9.40.UC1 and 9.40.TC1 of IDS which doesn't return the display labels correctly in combination with java.sql.CallableStatements. Therefore we can only show how the DADX syntax should look like, but we have not been able to test it.

This issue will be fixed in IDS 9.40.UC2 and 9.40.TC2.

To give you already the information on how the DADX syntax for an UDR what a result set should look like, take a look at the SPL UDR in Example 9-9 and the associated DADX syntax in Example 9-10. Notice the display label syntax in the stored procedure (`returning ... as ...`) and also the result_set definition and usage in the DADX file fragment.

Example 9-9 IDS 9 stored procedure with display labels for the result set

```
create procedure read_address  (lastname char(15))
         returning char(15) as pfname, char(15) as plname,
                  char(20) as paddress1, char(15) as pcity,
                  char(2)  as pstate, char(5) as pzipcode;
   define p_fname, p_city char(15);
   define p_add char(20);
   define p_state char(2);
   define p_zip char(5);
   select fname,  address1, city, state, zipcode
      into p_fname,  p_add, p_city, p_state, p_zip
      from customer
      where lname = lastname;
   return p_fname, lastname, p_add, p_city, p_state, p_zip;
end procedure;
```

Example 9-10 DADX syntax fragment for the UDR in Example 9-9

```
<dadx:result_set_metadata name="customerAddress" rowName="customer1">
   <dadx:column name="pfname" type="VARCHAR" nullable="true" />
   <dadx:column name="plname" type="VARCHAR" nullable="true" />
   <dadx:column name="paddress1" type="VARCHAR" nullable="true" />
   <dadx:column name="pcity" type="VARCHAR" nullable="true" />
   <dadx:column name="pstate" type="VARCHAR" nullable="true" />
   <dadx:column name="pzipcode" type="VARCHAR" nullable="true" />
</dadx:result_set_metadata>

<dadx:operation name="readOneCustomer">
   <dadx:documentation xmlns="http://www.w3.org/1999/xhtml">
   </dadx:documentation>
   <dadx:call>
      <dadx:SQL_call>
<![CDATA[
              { call read_address (:lname) }
]]>
      </dadx:SQL_call>
      <dadx:parameter name="lname" type="xsd:string" kind="in"/>
      <dadx:result_set name="customer1" metadata="customerAddress" />
   </dadx:call>
</dadx:operation>
```

DADX Web service deployment to a standalone WebSphere V5 server

First, you need to export your InformixXMLDemoEAR file to the file system by following the steps below:

1. Before you export your EAR file, verify the **group.properties** file in the **InformixXMLDemo/Java Source/groups.ITSOCustomerService** folder if the JDBC connect information are correct for the deployment server.
2. Open the **J2EE perspective** and switch to the **J2EE Navigator** window.
3. Right-click the **InformixMLDemoEAR** folder and select **Export** --> **EAR file**. Click **Next**.
4. In the **EAR Export** window use the **Browse** button to select an export directory in your file system. You can also select all three Options (**Export source files**, **Overwrite existing files without warning,** and **Include project build paths and meta-data files**) if required. Click **Finish**.
5. Now you can install the exported **InformixXMLDemoEAR.ear** file in WAS V5 by following the guidelines in Chapter 5 of this redbook.
6. As soon as the Web service has been installed in the deployment WebSphere server you can re-generate the WSDL file by using the following: **http://<hostname>:<portnumber>/InformixXMLDemo/ITSOCustomerService/ITSOCustomerService.dadx/WSDL** and save it into the **InformixXMLDemo/wsdl/ITSOCustomerService** folder if required.

9.3 Using IDS 9 as a Web service consumer

In the previous sections we described in detail how to use the WebSphere product family to enable IBM Informix IDS as a Web service provider.

Now we would like to focus on IDS as a Web service consumer.

This section is intended as a how-to-guide to use IDS 9 as a Web service consumer. It requires a basic knowledge of the Java language, for example you should know how to edit and compile a Java program. You should also have a basic understanding of the IDS 9 extensibility features.

Why IDS as a Web service consumer?

In addition to provide Web services, it can be very interesting for an application developer to integrate existing Web services. Those Web services could be either special B2B scenarios or public accessible services like currency conversion, stock ticker information, news, weather forecasts, search engines, and many more. Wouldn’t it be great to have dynamic access to an official currency conversion service on a database level if the application needs to deal with this information? Or if an application wants to relate actual business data stored in an IDS database against news from news agencies?

Sources for public accessible Web services are, for example:

http://www.webservicelist.com
http://www.xmethods.net

Web services rely on very simple open standards like XML and SOAP and be accessed through any kind of client application. Typically those applications are written in Java, C++, or C#. For somebody who already has an existing application which is based on an SQL database and also already utilizes business logic in the database server through user defined routines, developers might want to integrate access to Web services on the SQL level.

Some of the advantages of having Web services accessible from SQL would include easy access through the SQL language and standardized APIs (for example, ODBC, JDBC), moving the Web service results closer to the data processing in the database server which could speed up applications, and providing Web service access to the non Java or C++ developers.

What are the basic Web service consumer requirements for IDS?

In order to be able to call a Web service from within IDS, you need to be able to:

- Construct a SOAP message based on a given Web service description and
- Send this SOAP message to the Web service provider via the required protocol (typically HTTP)
- Finally, be able to receive the Web service response, parse it, and handle the results on an SQL level.

All of this needs to be executed from the IDS SQL layer to achieve the required portability.

Why IDS 9 and not IDS 7?

Although IDS 7 supports stored procedures with an already very powerful stored procedure language (SPL), it is somewhat limited if there is a need, for example, to access external networks or include external libraries.

IDS 9 through its very powerful DataBlade technology allows the easy integration of external routines written in C or Java into so called user defined routines (UDRs). Those UDRs can also be written in SPL. So one can say that UDRs are the generalized description of SPL, C, and Java stored procedures. In addition to the very flexible options of writing UDRs, IDS 9 also supports new data types and user defined types (UDTs).

Having these extensibility technologies available in IDS 9 in combination with the underlying, proven, high-end OLTP architecture of IDS 7 makes it a perfect choice to develop some database extensions which will provide access to Web services across standard network protocols.

Since you have the choice as an IDS 9 developer to either use C or Java for the development of Web service consumer routines, you could either include, for example, a C based SOAP framework or a Java based SOAP framework in your final solution.

To be as much platform independent as possible, and also to give you a kick-start on this topic, we chose the Apache AXIS Java framework for the development of IDS 9 Web service consumer routines.

The Apache AXIS framework

So what is the Apache AXIS framework?

The Axis framework is a Java-based, open source implementation of the latest SOAP specification, SOAP 1.2, and SOAP with Attachments specification from the Apache Group. The following are the key features of this AXIS framework:

- **Flexible messaging framework**: Axis provides a flexible messaging framework that includes handlers, chain, serializers, and deserializers. A handler is an object processing request, response, and fault flow. A handler can be grouped together into chains and the order of these handlers can be configured using a flexible deployment descriptor.
- **Flexible transport framework**: Axis provides a transport framework that helps you create your own pluggable transport senders and transport listeners.

- **Data encoding support**: Axis provides automatic serialization of a wide variety of data types as per the XML Schema specifications and provides a facility to use your own customized Serializer and Deserializer.
- **Additional features**: Axis provides full support for WSDL as well as Logging, Error, and Fault Handling mechanisms.

Axis also provides a simple tool set to easily generate Java classes based on given Web service description files (WSDL) and has tools to monitor Web services.

The latest Axis distribution and more detailed information about Axis can be obtained at:

```
http://ws.apache.org/axis
```

9.3.1 IDS 9 and J/Foundation

In this section we present an overview of these topics.

Overview

IDS 9 with J/Foundation enables database developer's to write server-side business logic using the Java language. Java User Defined Routines (UDRs) have complete access to the leading extensible database features of the IDS 9 database. Making IDS 9 the ideal platform for Java database development.

In addition to Java UDRs, IDS conforms to the SQLJ standard for Java-stored procedures, enabling the use of the standard Java packages that are included in the Java Development Kit (JDK). Writing UDRs in Java delivers far more flexible applications that can be developed faster than C, and more powerful and manageable than stored procedure languages.

IDS with J/Foundation provides these advantages over other Java based solutions:

- Better performance and scalability
- Fully certified and optimized standard JVMs for each supported platform
- Simpler application integration
- Easy migration of existing Java applications
- Transaction control through stored data

J/Foundation is provided with IDS on many of the supported IDS 9.40 platforms. Supported platforms include Sun Solaris 32 bit, HP-UX 32 bit, Microsoft Windows 2000, Windows 2003, Window XP, Linux, IBM AIX, SGI Irix, and Compaq Tru 64.

Technology

IDS 9 provides the infrastructure to support Java UDRs. The database server binds SQL UDR signatures to Java executables and provides mapping between SQL data values and Java objects so that the database server can pass parameters and retrieve returned results. IDS 9 also provides support for data type extensibility and sophisticated error handling.

Java UDRs execute on specialized virtual processors called Java Virtual Processors (JVPs). IDS 9 embeds a Java Virtual Machine (JVM) in the code of each JVP. The JVPs are responsible for executing all server-based Java UDRs and applications.

Although the JVPs are mainly used for Java-related computation, they have the same capabilities as a CPU VP, and they can process all types of SQL queries. This eliminates the need to ship Java-related queries back and forth between CPU VPs and JVPs.

For more technical details of J/Foundation, refer to the IBM Informix J/Foundation Developer’s Guide.

9.3.2 Installation and configuration of IDS 9 and AXIS for the examples

> **Tip:** All of the configuration and installation information in this section is based on Windows 2000, but can be easily also applied to other platforms like Linux or UNIX.

AXIS installation and preparation

First, download the AXIS release from `http://ws.apache.org/axis/releases.html` — the release we have been using for the examples below is based on AXIS release 1.1rc2. After downloading the release, extract the AXIS distribution into a directory of your choice (for example, directly into the C:\ directory). Make sure that you also extract the folder structure. If you are finished, you should have an <install_dir>\axis-1_1RC2 directory.

In addition to AXIS we also need a JAXP 1.1 XML compliant parser. The recommended one is the Apache Xerces: Just download the latest stable version from `http://xml.apache.org/dist/xerces-j` (for example, Xerces-J-bin.2.4.0.zip) and extract it into a local directory (for example, C:\). Eventually you should have an <install_dir>\xerces-2_4_0 directory.

IDS 9 with J/Foundation configuration for AXIS

Since the AXIS Framework is Java based, we need to configure IDS 9 for Java UDRs. Before we go ahead, make sure that you're using an IDS 9 with J/Foundation. You can verify this by checking the $INFORMIXDIR/extend directory for the existence of a **krakatoa** subdirectory. If this directory is missing you don't have the correct version of IDS.

First, we need to enable J/Foundation for your IDS 9 instance:

1. Create an sbspace to hold the Java JAR files. The database server stores Java JAR files as smart large objects in the system default sbspace. If you do not already have a default sbspace, you must create one. After you create the sbspace, set the SBSPACENAME configuration parameter in the ONCONFIG file to the name that you gave to the sbspace.
2. Add (or modify) the Java configuration parameters in the ONCONFIG configuration file. The ONCONFIG configuration file ($INFORMIXDIR/etc/$ONCONFIG) includes the following configuration parameters that affect Java code:
   - JDKVERSION
   - JVPPROPFILE
   - JVMTHREAD
   - JVPCLASSPATH
   - JVPHOME
   - JVPJAVALIB
   - JVPJAVAVM
   - JVPLOGFILE
   - JVPARGS
   - VPCLASS

   Make sure that these parameters exist or are not un-commented. For an example ONCONFIG file fragment, see Example 9-11.

Example 9-11 J/Foundation settings for the AXIS framework

```
VPCLASS        jvp,num=1  # Number of JVPs to start with

JVPJAVAHOME     C:\informix\extend\krakatoa\jre# JDK installation root directory
JVPHOME         C:\informix\extend\krakatoa# Krakatoa installation directory

JVPLOGFILE      C:\informix\extend\krakatoa\ol_jvp.log# VP log file
```

```
JVPPROPFILE     C:\informix\extend\krakatoa\.jvpprops_alexk# JVP property file

JDKVERSION      1.3       # JDK version supported by this server

# The path to the JRE libraries relative to JVPJAVAHOME
JVPJAVALIB      \bin\

JVPJAVAVM       hpi;server;verify;java;net;zip;jpeg

# Classpath to use upon Java VM start-up (use _g version for debugging)
JVPCLASSPATH    file:C:\informix\extend\krakatoa\jvp_classpath

JVPARGS   -Djava.security.policy=C:\informix\extend\krakatoa\informix.policy
```

In the foregoing example, we also define the JVPCLASSPATH to point to a file in the krakatoa directory. Having an external file to contain the JVP classpath information gives us more flexibility regarding the maximal length of the JVPCLASSPATH since the length in the ONCONFIG file is otherwise limited to 256 characters. See Example 9-12 for an AXIS compliant classpath file.

Tip: In our examples we're copying the AXIS class libraries directly into the $INFORMIXDIR\extend\krakatoa directory to avoid any changes to the informix.policy file. I would be probably a cleaner approach to keep the AXIS files in their original directories and adjust the informix.policy file to allow access for the J/Foundation class loader.

Example 9-12 jvp_classpath file for the AXIS integration

```
C:\informix\extend\krakatoa\krakatoa.jar;C:\informix\extend\krakatoa\jdbc.jar;C:\informix\e
xtend\krakatoa\axis.jar;C:\informix\extend\krakatoa\jaxrpc.jar;C:\informix\extend\krakatoa\
saaj.jar;C:\informix\extend\krakatoa\commons-logging.jar;C:\informix\extend\krakatoa\common
s-discovery.jar;C:\informix\extend\krakatoa\wsdl4j.jar;C:\informix\extend\krakatoa\xercesIm
pl.jar;C:\informix\extend\krakatoa\xmlParserAPIs.jar;C:\informix\extend\krakatoa\axis-ant.j
ar;C:\informix\extend\krakatoa\log4j-1.2.4.jar;
```

In addition, we also need to modify the default security settings for the Java VM.

Tip: Normally you only need to define the security policies in an informix.policy file in the JVPHOME directory. Due to a minor bug in IDS 9.40UC1 and 9.40.TC1 it is not being read by the database server. The workaround is to define the JVPARGS ONCONFIG parameter to point to a policy file (as in Example 9-11). The policy file for the usage with AXIS should look like Example 9-13.

This problem will be fixed in IDS 9.40.UC2 and IDS 9.40.TC2.

Example 9-13 The informix.policy file with AXIS support

```
grant codeBase "file:/C:/informix/extend/krakatoa/-" {
   permission java.security.AllPermission;
};

grant {
   permission java.io.SerializablePermission "enableSubstitution";
   permission java.lang.RuntimePermission "shutdownHooks";
   permission java.lang.RuntimePermission "setContextClassLoader";
   permission java.lang.RuntimePermission "reflectionFactoryAccess";
   permission java.lang.RuntimePermission "unsafeAccess";
   permission java.net.NetPermission "specifyStreamHandler";
   permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
   permission java.util.PropertyPermission "user.language","write";
   permission java.util.PropertyPermission "user.dir","write";
   permission java.security.SecurityPermission "getPolicy";
   permission java.util.PropertyPermission "java.naming.factory.initial","write";
   permission java.util.PropertyPermission "java.naming.provider.url","write";
};

grant {
   permission java.util.PropertyPermission "java.protocol.handler.pkgs","write";
};
```

3. Create the JVP properties file (optional). It is optional to define the JVP properties, but they are often used for debugging Java UDRs. You will find a template file in the $INFORMIXDIR\extend\krakatoa directory.
4. Set environment variables.You do not need any extra environment variables to execute UDRs written in Java code. However, since we are developing Java UDRs, you must include JVPHOME/krakatoa.jar in your CLASSPATH environment variable so that JDK can compile the Java source files that use Informix Java packages. For a complete description of the CLASSPATH settings for AXIS UDR development, refer to “Java classpath settings for AXIS UDR development” on page 238.
5. Now copy all Java class libraries from the AXIS distribution (for example, c:\axis-1_1RC2\lib) into the $INFORMIXDIR\extend\krakatoa directory.
6. Finally, copy the **xercesImpl.jar** and the **xmlParserAPIs.jar** class library from the Xerces distribution (for example, C:\xerces-2_4_0) also into the $INFORMIXDIR\extend\krakatoa directory.

Java classpath settings for AXIS UDR development

The Java classpath for developing the AXIS based UDRs should look like the one in Example 9-14.

Example 9-14 Classpath settings for AXIS UDR development

```
C:\axis-1_1RC2\lib\axis.jar;C:\axis-1_1RC2\lib\jaxrpc.jar;C:\axis-1_1RC2\lib\saaj.jar;c:\ax
is-1_1RC2\lib\commons-logging.jar;C:\axis-1_1RC2\lib\commons-discovery.jar;C:\axis-1_1RC2\l
ib\wsdl4j.jar;C:\xerces-2_4_0\xercesImpl.jar;C:\xerces-2_4_0\xmlParserAPIs.jar;C:\infor
mix\extend\krakatoa\krakatoa.jar;.
```

9.3.3 The basic IDS Web service consumer development steps

Before we start to access some Web services from IDS 9, let’s consider the required steps:

1. Obtain access to the WSDL file for the desired Web service, either by downloading it to the local server or have access to it via the http protocol.
2. Use the AXIS WSDl2Java tool to generate the Web service Java class files.
3. Compile the class files from step 2 (no coding needed!)
4. Write a small Java UDR wrapper to access the generated AXIS classes. You can take the Java UDR wrappers from the examples below as templates for your own projects.
5. Create a Java jar file which should contain the generated AXIS class files and your Java UDR wrapper class.
6. Write a simple SQL script to register your Java UDR in the IDS database of your choice.
7. Register your Java UDR in the database of your choice with the SQL script from step 6.
8. Run and test your Java UDRs to access the Web services.

9.3.4 The AXIS WSDL2Java tool

The WSDL2Java tool which part of the org.apache.axis.wsdl.WSDL2Java class is the starting point to generate Java classes from a given WSDL file.

Its normally being executed by the following command line:

```
java org.apache.axis.wsdl.WSDL2Java <WSDL-file-URL>
```

Tip: To make the execution of this tool easier for you throughout the examples in the following sections, we suggest to create a small batch/script file like the one shown in Example 9-15. Call this file (in a Windows environment) wsdl2java.bat.

Example 9-15 The wsdl2java.bat file (for Windows platforms)

```
@echo off
SET TMPCLASSPATH=%CLASSPATH%
SET CLASSPATH=.
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\axis.jar
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\jaxrpc.jar
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\saaj.jar
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\commons-logging.jar
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\commons-discovery.jar
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\wsdl4j.jar
SET CLASSPATH=%CLASSPATH%;C:\xerces-2_4_0\xercesImpl.jar
SET CLASSPATH=%CLASSPATH%;C:\xerces-2_4_0\xmlParserAPIs.jar
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\axis-ant.jar
SET CLASSPATH=%CLASSPATH%;C:\axis-1_1RC2\lib\log4j-1.2.4.jar
echo ---------------------------------------------
echo --= Classpath has been set for AXIS needs =--
echo ---------------------------------------------
java org.apache.axis.wsdl.WSDL2Java -p %2 -v %1
SET CLASSPATH=%TMPCLASSPATH%
```

The wsdl2java.bat script file has two parameters: the WSDL file URL and a package name. The package name becomes also a local subdirectory to the directory in which you're executing the wsdl2java.bat file. The WSDL file URL can be either a local filename or an URL on the Internet (for example, http://www.someserver.com/webserviceinfo/myservice.wsdl).

9.3.5 A simple IDS Web service example — Currency Exchange project

So let's start with our first example project, the currency exchange Web service from http://www.xmethods.net — this Web service allows the currency conversion between different foreign currencies. You only have to provide the source currency country name and then the target currency country name.

Now follow the development steps we have outlined in 9.3.3, "The basic IDS Web service consumer development steps" on page 238:

1. Obtain a copy of the Web service WSDL file:

   The WSDL file for this Web service can be obtained from http://www.xmethods.net/sd/2001/CurrencyExchangeService.wsdl — You can either download the WSDL file to your local disk or use the above URL directly as input to the WSDL2Java tool. For your convenience we have also included the WSDL file in Example 9-16.

Example 9-16 The CurrencyExchange WSDL file

```
<?xml version="1.0"?>
<definitions name="CurrencyExchangeService"
targetNamespace="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl"
xmlns:tns="http://www.xmethods.net/sd/CurrencyExchangeService.wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
   <message name="getRateRequest">
      <part name="country1" type="xsd:string"/>
      <part name="country2" type="xsd:string"/>
   </message>
   <message name="getRateResponse">
      <part name="Result" type="xsd:float"/>
   </message>
   <portType name="CurrencyExchangePortType">
      <operation name="getRate">
         <input message="tns:getRateRequest" />
         <output message="tns:getRateResponse" />
      </operation>
   </portType>
   <binding name="CurrencyExchangeBinding"
type="tns:CurrencyExchangePortType">
      <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="getRate">
         <soap:operation soapAction=""/>
         <input >
            <soap:body use="encoded"
namespace="urn:xmethods-CurrencyExchange"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </input>
         <output >
            <soap:body use="encoded"
namespace="urn:xmethods-CurrencyExchange"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
         </output>
      </operation>
   </binding>
   <service name="CurrencyExchangeService">
      <documentation>Returns the exchange rate between the two
currencies</documentation>
      <port name="CurrencyExchangePort"
binding="tns:CurrencyExchangeBinding">
         <soap:address location="http://services.xmethods.net:80/soap"/>
      </port>
   </service>
</definitions>
```

While looking at the WSDL file, you might have already noticed that the two input parameter (country1 and country2) are of type String and the result is of type float.

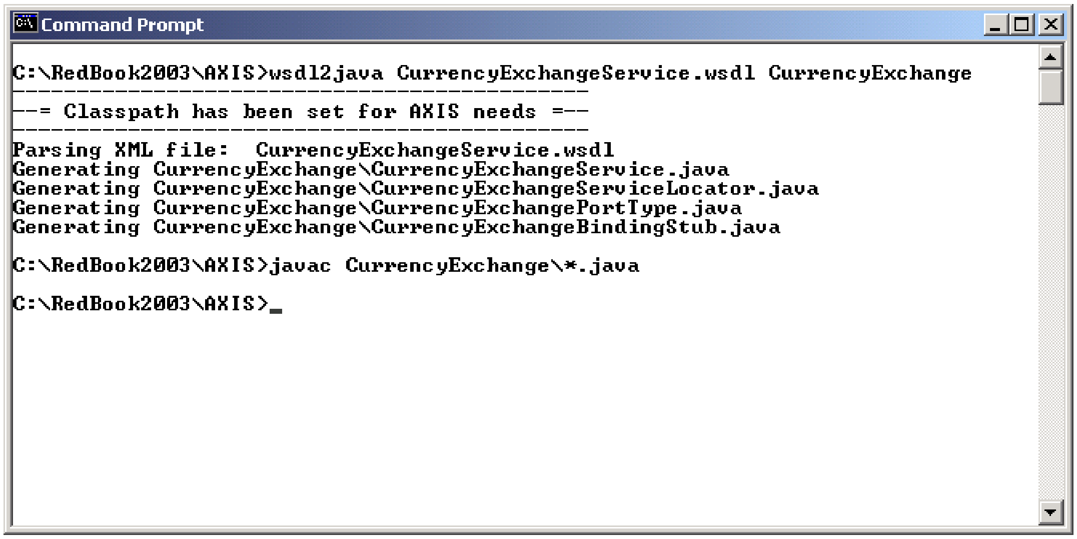
2. Now we need to generate the AXIS Java classes for our Web service.

   To do this, create a directory of your choice (for example, C:\Redbook2003\AXIS) and copy the WSDL file into this directory.

   From a command line window, run the prepared wsdl2java.bat scrip file with the following options:

   ```
   wsdl2java CurrencyExchangeService.wsdl CurrencyExchange
   ```

This will generate a subdirectory called CurrencyExchange, and this subdirectory should contain the following files: CurrencyExchangeBindingStub.java, CurrencyExchangePortType.java, CurrencyExchangeService.java, CurrencyExchangeServiceLocator.java (Figure 9-12).



```
C:\RedBook2003\AXIS>wsdl2java CurrencyExchangeService.wsdl CurrencyExchange
--------------------------------------------
--= Classpath has been set for AXIS needs =--
--------------------------------------------
Parsing XML file:  CurrencyExchangeService.wsdl
Generating CurrencyExchange\CurrencyExchangeService.java
Generating CurrencyExchange\CurrencyExchangeServiceLocator.java
Generating CurrencyExchange\CurrencyExchangePortType.java
Generating CurrencyExchange\CurrencyExchangeBindingStub.java

C:\RedBook2003\AXIS>javac CurrencyExchange\*.java

C:\RedBook2003\AXIS>_
```

Figure 9-12 Generating and compiling the CurrencyExchange AXIS classes

3. Now you need to compile the generated Java classes from step 2 by simply executing:

   ```
   javac CurrencyExchange\*.java
   ```

   Before you execute the Java compiler, make sure that you have set the CLASSPATH environment variable correctly (see Example 9-14) and also that you have the Java compiler in your PATH environment variable (for example, C:\jdk1.4.0\bin).

4. In order to utilize the generated AXIS class files for the CurrencyExchange Web service we need to write a simple Java wrapper UDR to call the required methods.

   So first take a look at the final code in Example 9-17.

Example 9-17 CurrencyExchangeUDRs..java

```
import CurrencyExchange.*;

public class CurrencyExchangeUDRs
{
   public static double currencyExchange( String country1, String country2)
                                                                  throws Exception
   {
      double  RetVal;

      CurrencyExchange.CurrencyExchangeService service =
                        new CurrencyExchange.CurrencyExchangeServiceLocator();

      CurrencyExchange.CurrencyExchangePortType port =
                                             service.getCurrencyExchangePort();

      RetVal = port.getRate(country1, country2);

      return RetVal;
   }
};
```

The currencyExchange method implements the Web service API by accepting the two country descriptions as Java strings and returns a Java double type.

First, we need to create a service instance of type CurrencyExchangeService which can be achieved by creating a new CurrencyExchangeServiceLocator object.

Then we need to obtain the port object of type CurrencyExchangePortType from the service object.

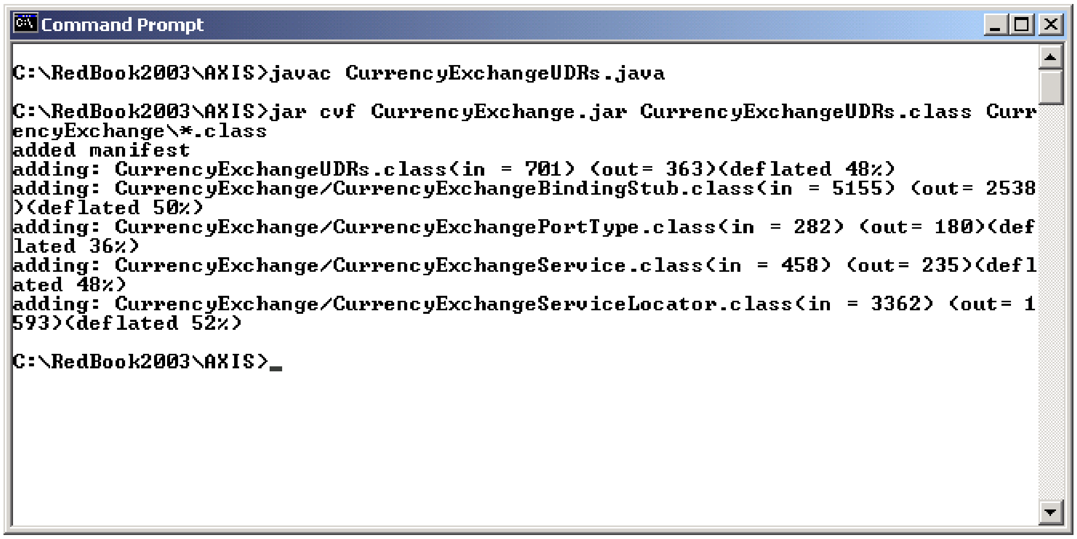
And finally we need to call the getRate(String, String) method to generate the SOAP message which is then being sent to the Web service provider.

The getRate() method is defined in the CurrencyExchangeBindingStub.java file.

Save the Java code from Example 9-17 into your example directory (for example, C:\RedBook2003\AXIS) as CurrencyExchangeUDRs.java.

Now compile the CurrencyExchangeUDRs.java file:

```
javac CurrencyExchangeUDRs.java
```



```
C:\RedBook2003\AXIS>javac CurrencyExchangeUDRs.java

C:\RedBook2003\AXIS>jar cvf CurrencyExchange.jar CurrencyExchangeUDRs.class Curr
encyExchange\*.class
added manifest
adding: CurrencyExchangeUDRs.class(in = 701) (out= 363)(deflated 48%)
adding: CurrencyExchange/CurrencyExchangeBindingStub.class(in = 5155) (out= 2538
)(deflated 50%)
adding: CurrencyExchange/CurrencyExchangePortType.class(in = 282) (out= 180)(def
lated 36%)
adding: CurrencyExchange/CurrencyExchangeService.class(in = 458) (out= 235)(defl
ated 48%)
adding: CurrencyExchange/CurrencyExchangeServiceLocator.class(in = 3362) (out= 1
593)(deflated 52%)

C:\RedBook2003\AXIS>_
```

Figure 9-13 Compile the UDR wrapper and create the jar file

5. In preparation for the registration in your IDS 9 database we need to pack all of our classes (generated AXIS classes plus the UDR wrapper) into a Java jar file. To do this, execute this command:

   ```
   jar cvf CurrencyExchange.jar CurrencyExchangeUDRs.class CurrencyExchange\*.class
   ```

   (Also see the foregoing Figure 9-13.)

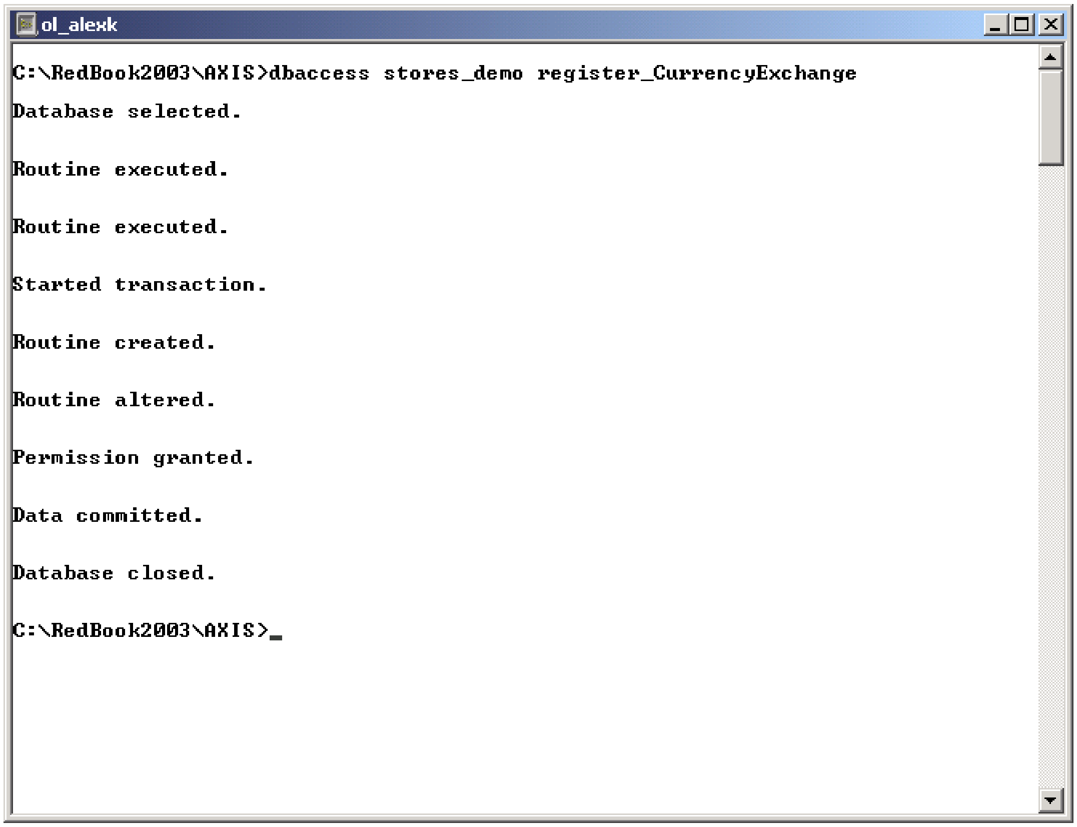
6. Now we need to create a simple SQL script to first store our CurrencyExchange.jar file which contains the UDR wrapper plus the generated AXIS classes into the database and then connect the Java classes with the SQL layer by defining a Java UDR with the CREATE FUNCTION SQL statement.

   You can use the SQL script from Example 9-18 as a template for similar Java UDRs in the future. So on the SQL level we're naming our user defined routine simply CurrencyExchange. This routine takes two LVARCHARs as parameters and returns a SQL FLOAT data type which matches the Java double type.

Example 9-18 The register_CurrencyExchange.sql script

```
execute procedure
install_jar('file:C:/RedBook2003/AXIS/CurrencyExchange.jar','CurrencyExchange');

execute procedure ifx_allow_newline('t');

begin work;

create function CurrencyExchange (lvarchar, lvarchar)
returns float as exchange_rate
external name 'CurrencyExchange:CurrencyExchangeUDRs.currencyExchange(java.lang.String,
java.lang.String)'
   language java;

alter function CurrencyExchange (lvarchar, lvarchar)
   with (add parallelizable);

grant execute on function CurrencyExchange (lvarchar, lvarchar) to public;

commit work;
```

The install_jar procedure stores the CurrencyExchange.jar into a smart blob in the default smart blob space in the IDS 9 instance and gives it the symbolic name CurrencyExchange. which can be used in the create function statement to reference the jar file. See Figure 9-14.



```
C:\RedBook2003\AXIS>dbaccess stores_demo register_CurrencyExchange
Database selected.

Routine executed.

Routine executed.

Started transaction.

Routine created.

Routine altered.

Permission granted.

Data committed.

Database closed.

C:\RedBook2003\AXIS>_
```

Figure 9-14 Register the Java UDR with stores_demo database

The create function finally registers the Java UDR with the database and makes it available to any SQL compliant application.

7. In order to register your CurrencyExchange UDR you should have a database with logging enabled. Assuming you might want to register your UDR with the IDS stores_demo database you only have to run the SQL script by executing:

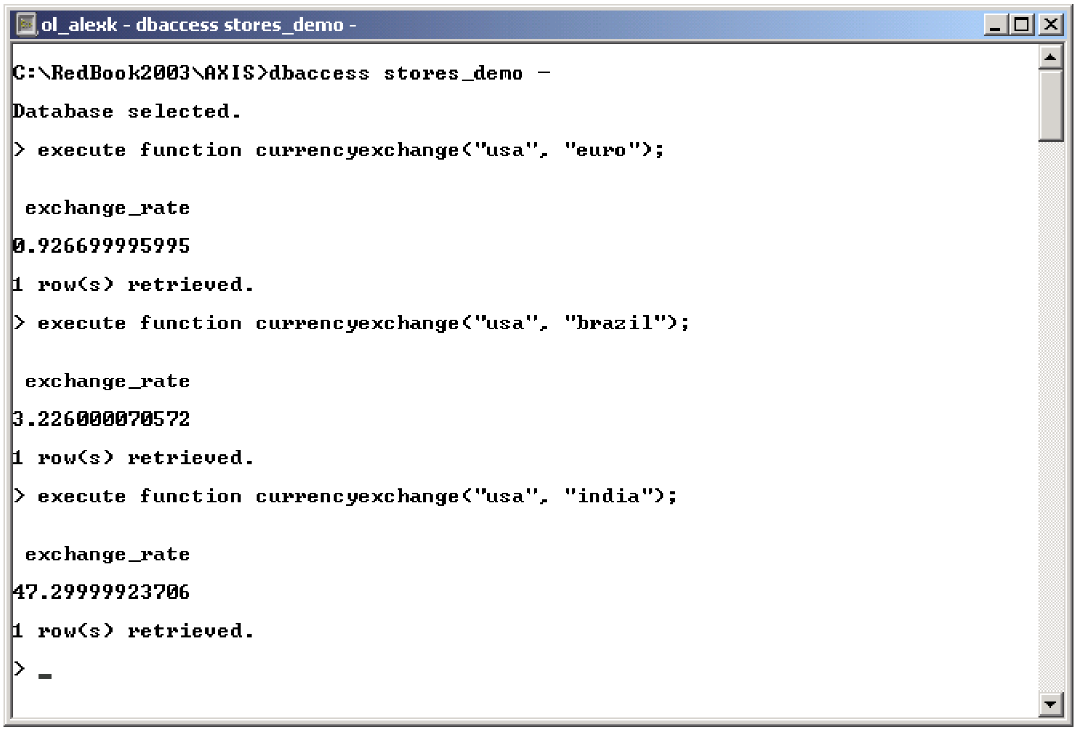
   ```
   dbaccess stores_demo register_CurrencyExchange
   ```

   See also Figure 9-14.

8. Now we're ready to test the Java UDR to call the Web service. Before you can test the UDR make sure that you're connected to the Internet. Then, for example, start dbaccess to connect to the stores database and execute the CurrencyExchange function. Since we're using SQL and SQL doesn't differentiate between lowercase and uppercase letters we just simply type:

   ```
   execute function currencyexchange(“<country1>”, “<country2”>).
   ```

   See Figure 9-15 for some sample values and results.



```
C:\RedBook2003\AXIS>dbaccess stores_demo -

Database selected.

> execute function currencyexchange("usa", "euro");


 exchange_rate

0.926699995995

1 row(s) retrieved.

> execute function currencyexchange("usa", "brazil");


 exchange_rate

3.226000070572

1 row(s) retrieved.

> execute function currencyexchange("usa", "india");


 exchange_rate

47.29999923706

1 row(s) retrieved.

> _
```

Figure 9-15 Test of the currencyexchange Java UDR

For additional valid values for the country parameters, consult the CurrencyExchange Web service description on the Web site:

```
http://www.xmethods.net
```

Tip: If you are behind a firewall, then you might have to set additional properties for the J/Foundation Java VM via the **JVPARGS** variable.

So if you typically use a socks compliant proxy, replace the **JVPARGS** value in the ONCONFIG file with the following line:

```
-Djava.security.policy=C:\informix\extend\krakatoa\informix.policy;-DsocksProxyHost=
<SocksProxyhostname>;-DsocksProxyPort=<socksProxyPortvalue>
```

If you're using a standard HTTP proxy, you might have to use the following value for **JVPARGS** instead:

```
-Djava.security.policy=C:\informix\extend\krakatoa\informix.policy;-Dhttp.proxyHost=
<httpProxyhostname>;-Dhttp.proxyPort=<httpProxyPortvalue>
```

For more details about proxy support in the Java VM, consult this Web site:

http://java.sun.com/j2se/1.4.1/docs/guide/net/properties.html

9.3.6 A complex IDS Web service example — Google search

Since we have now learned the basic steps to use IDS 9 as a Web service consumer, let's take a look at a slightly more complex example.

This time we are going to implement a Java UDR which accesses the Google search Web service API. In order to be able to use that service you need to register at the Google API Web site at http://www.google.com/apis to obtain a license key which allows up to 1000 Google searches a day via their Web service interface. In a second preparation step you should download the Google API developer's kit which also contains the WSDL file for their Web service. Due to its length we haven't included this WSDL as an example in the redbook.

The Google Web service provides three different callable methods:

- doGoogleSearch (key, q, start, maxResults, filter, restrict, safeSearch, lr, ie, oe)
  - Returns doGoogleSearchResponse
- doGetCachedPage (key, url)
  - Returns doGetCachedPageResponse
- doSpellingSuggestion (key, phrase)
  - Returns doSpellingSuggestionResponse

For the following example we're just implementing the first and most important method: doGoogleSearch.

Let's take a brief look at the parameters of the doGoogleSearch function:

Table 9-1 doGoogleSearch parameters

Parameter name	Type	Function
key	string	License key from Google
q	string	Search string
start	int	Start result (begins with 0)
maxResults	int	Number of results per query. Maximum is 10.
filter	boolean	Automatic Google search results filtering

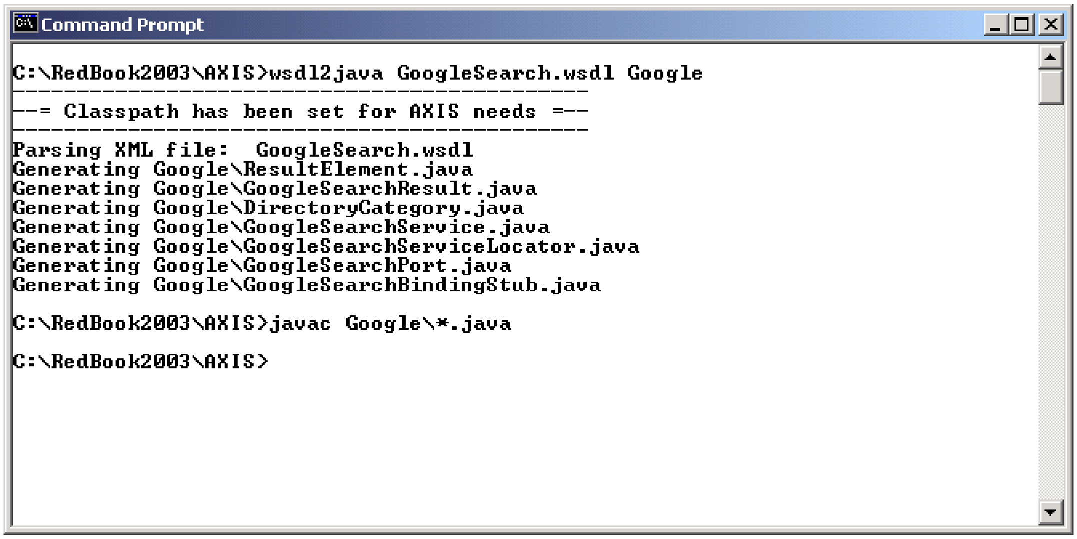
Parameter name	Type	Function
restrict	string	Restricts the search to a subset of the Google Web index.
safeSearch	boolean	Enables filtering of adult content in the search results
lr	string	Restricts the search to documents within one or more languages.
ie	string	This parameter has been deprecated and is ignored
oe	string	This parameter has been deprecated and is ignored

Out of those parameters we choose q, start, restrict and lr for the Java wrapper UDR which we will name GoogleSearch. Since the GoogleSearch UDR needs to able to return multiple resultsets back to the caller we need to write an iterator UDR in Java.

What is an iterator function? An iterator function returns an active set of items. Each iteration of the function returns one item of the active set. Such a function is normally associated with a cursor in the calling program. It can be also executed within an SQL script by just do an EXECUTE FUNCTION call or in combination with an INSERT statement. An iterator function is similar to an SPL function that contains the RETURN WITH RESUME statement. For more details on iterator functions and how to implement them in Java, take a look at the IBM Informix Creating User-Defined Routines and User-Defined Data Types user guide and the IBM Informix J/Foundation Developer's Guide.

For our code example we assume that we do have a directory C:\RedBook2003\AXIS which already contains the GoogleSearch.wsdl file. Now we have to follow again the eight simple development steps to create our GoogleSearch() Java UDR:

1. We can already skip step 1, since we already obtained and copied the GoogleSearch.wsdl file into our working directory.



```
C:\RedBook2003\AXIS>wsdl2java GoogleSearch.wsdl Google
---------------------------------------------
--= Classpath has been set for AXIS needs =--
---------------------------------------------
Parsing XML file:  GoogleSearch.wsdl
Generating Google\ResultElement.java
Generating Google\GoogleSearchResult.java
Generating Google\DirectoryCategory.java
Generating Google\GoogleSearchService.java
Generating Google\GoogleSearchServiceLocator.java
Generating Google\GoogleSearchPort.java
Generating Google\GoogleSearchBindingStub.java

C:\RedBook2003\AXIS>javac Google\*.java

C:\RedBook2003\AXIS>
```

Figure 9-16 Generate and compile the AXIS classes for the Google search service

2. From a command line window, run the prepared wsdl2java.bat scrip file with the options:

   ```
   wsdl2java GoogleSearch.wsdl Google
   ```

   This will generate a subdirectory called Google and this subdirectory should contain the following files: DirectoryCategory.java, GoogleSearchBindingStub.java, GoogleSearchPort.java, GoogleSearchResult.java, GoogleSearchService.java, GoogleSearchServiceLocator.java, ResultElement.java.

3. Now you need to compile the generated Java classes from step 2 by simply executing

   ```
   javac Google\*.java
   ```

   Before you execute the Java compiler, make sure that you have set the CLASSPATH environment variable correctly (see Example 9-14) and also that you have the Java compiler in your PATH environment variable (for example, C:\jdk1.4.0\bin).

4. Now we need to write the Java wrapper UDR to call the AXIS methods. The procedure is basically the same to the simple example in the previous section, except that we need to write a slightly more complex iterator function. Use the following code in Example 9-19 as a template for similar routines.

Example 9-19 The GoogleUDRs.java file

```
import Google.*;
import com.informix.udr.UDRManager;
import com.informix.udr.UDREnv;
import com.informix.udr.UDRLog;

public class GoogleUDRs
{

   public int max_results;
   public int counter;
   public Google.ResultElement[] results;

   public static String GoogleSearch
         (String searchstring, int start, String language,
                                 String country_topic) throws Exception
   {
      String New_RetVal;

      UDREnv env = (UDREnv) UDRManager.getUDREnv();
      int iter = env.getSetIterationState();
      UDRLog log = (UDRLog) env.getLog();

      New_RetVal = new String();

      if (iter == UDREnv.UDR_SET_INIT)
      {
         GoogleUDRs state = new GoogleUDRs();

         state.counter = 0;
         state.max_results = 0;

         Google.GoogleSearchService service =
                                    new Google.GoogleSearchServiceLocator();

        Google.GoogleSearchPort port = service.getGoogleSearchPort();

        Google.GoogleSearchResult RetVal = (Google.GoogleSearchResult)
                port.doGoogleSearch("<your_Google_license_key>",
                           searchstring, start, 10, true, country_topic,
                           false, language, null, null);

         state.results = RetVal.getResultElements();

         state.max_results = state.results.length;

         env.setUDRState(state);
         env.setSetIterationIsDone(false);
```

```
         return null;
      }
      else if (iter == UDREnv.UDR_SET_END)
      {
         env.setSetIterationIsDone(true);
         return null;
      }
      else if (iter == UDREnv.UDR_SET_RETONE)
      {
         GoogleUDRs state = (GoogleUDRs)env.getUDRState();

         if (state.counter < state.max_results)
         {
            New_RetVal = "ROW (\"" +
                      state.results[state.counter].getTitle() + "\",\"" +
                      state.results[state.counter].getSnippet() + "\",\"";
            New_RetVal = New_RetVal +
                      state.results[state.counter].getSummary() + "\",\"" ;
            New_RetVal = New_RetVal +
                      state.results[state.counter].getURL() + "\"";
            New_RetVal = New_RetVal + ")";

            state.counter++;

            env.setSetIterationIsDone(false);
            env.setUDRState(state);
            return New_RetVal;
         }
         else
         {
            env.setSetIterationIsDone(true);
            return null;
         }
      }
      return null;
   }
};
```

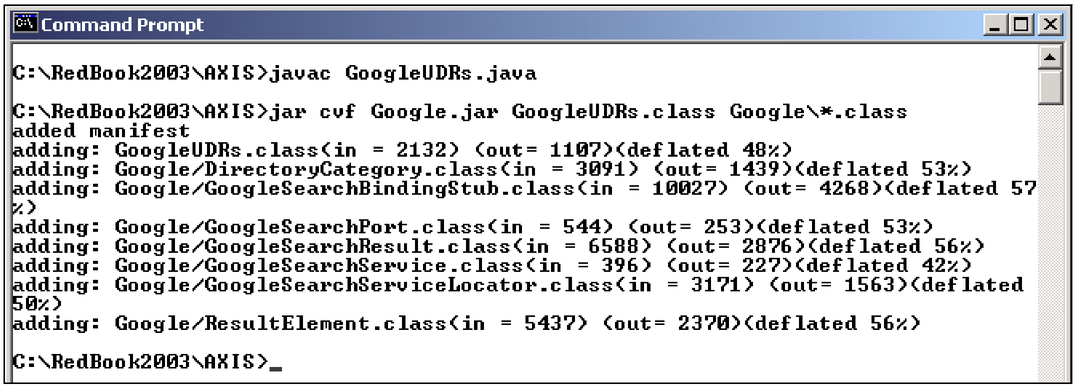
If you take a closer look at the code above you will notice that this iterator routine supports three states: an INIT state in which we initialize everything and also call the doGoogleSearch() method, an RETONE state which is being used to return one Google search result row at a time and finally an END state which is being called to do the typical housekeeping of an routine before we return the control to the caller.

In between every call of the routine, the state of the routine, including the row counter, for example, is being saved until the next call. This is being achieved by utilizing the `com.informix.udr.UDREnv and the com.informix.udr.UDRManager` packages which are part of J/Foundation.

Since we want to return multiple Google.ResultElement components we're also constructing the string representation of a ROW data type. Having a ROW data type available for the return results, makes its easier to later further process the result set, for example, by storing it into a table or similar.

Now save the code from Example 9-19 as GoogleUDRs.java into the build directory (for example, C:\RedBook2003\AXIS) and compile it by executing:

```
javac GoogleUDRs.java
```



```
C:\RedBook2003\AXIS>javac GoogleUDRs.java

C:\RedBook2003\AXIS>jar cvf Google.jar GoogleUDRs.class Google\*.class
added manifest
adding: GoogleUDRs.class(in = 2132) (out= 1107)(deflated 48%)
adding: Google/DirectoryCategory.class(in = 3091) (out= 1439)(deflated 53%)
adding: Google/GoogleSearchBindingStub.class(in = 10027) (out= 4268)(deflated 57
%)
adding: Google/GoogleSearchPort.class(in = 544) (out= 253)(deflated 53%)
adding: Google/GoogleSearchResult.class(in = 6588) (out= 2876)(deflated 56%)
adding: Google/GoogleSearchService.class(in = 396) (out= 227)(deflated 42%)
adding: Google/GoogleSearchServiceLocator.class(in = 3171) (out= 1563)(deflated
50%)
adding: Google/ResultElement.class(in = 5437) (out= 2370)(deflated 56%)

C:\RedBook2003\AXIS>_
```

Figure 9-17 Compile the GoogleUDRs.java file and create the Google.jar file

5. As in the previous example, we need to pack all of our classes (generated AXIS classes plus the UDR wrapper) into a Java jar file. To do this, execute this command:

   ```
   jar cvf Google.jar GoogleUDRs.class Google\*.class
   ```

   Also see the foregoing Figure 9-17.

6. For registration of the Java UDR with the database server we need (again) an SQL script. For the GoogleSearch() function, use the register_Google.sql file from Example 9-20.

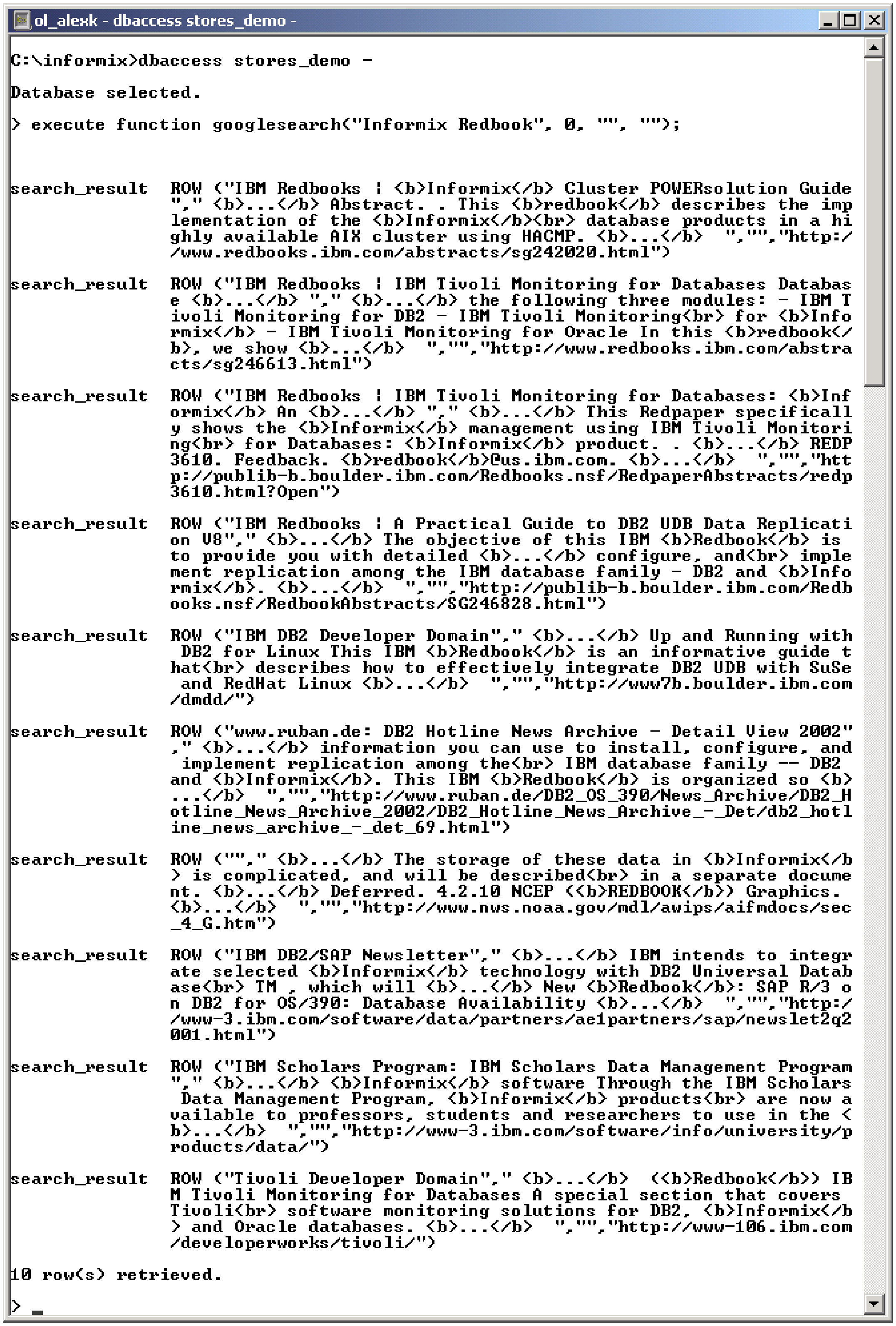
Example 9-20 The register_Google.sql script

```
execute procedure
            install_jar('file:C:/RedBook2003/AXIS/Google.jar','Google');

execute procedure ifx_allow_newline('t');

begin work;

create function GoogleSearch (lvarchar, integer, lvarchar, lvarchar)
returns lvarchar as search_result
external name 'Google:GoogleUDRs.GoogleSearch(java.lang.String, int,
                                          java.lang.String, java.lang.String)'
language java;

alter function GoogleSearch (lvarchar, integer, lvarchar, lvarchar)
   with (add parallelizable);

alter function GoogleSearch (lvarchar, integer, lvarchar, lvarchar)
   with (add not variant);

alter function GoogleSearch (lvarchar, integer, lvarchar, lvarchar)
   with (add iterator);

grant execute on function GoogleSearch (lvarchar, integer, lvarchar, lvarchar) to public;

commit work;
```

7. In order to register your CurrencyExchange UDR you should have a database with logging enabled. Assuming you might want to register your UDR with the IDS stores_demo database you only have to run the SQL script by executing:

   ```
   dbaccess stores_demo register_Google
   ```

   The output from dbaccess should be similar to the one in Figure 9-14.



```
C:\informix>dbaccess stores_demo -

Database selected.

> execute function googlesearch("Informix Redbook", 0, "", "");


search_result  ROW ("IBM Redbooks | <b>Informix</b> Cluster POWERsolution Guide
               "," <b>...</b> Abstract. . This <b>redbook</b> describes the imp
               lementation of the <b>Informix</b><br> database products in a hi
               ghly available AIX cluster using HACMP. <b>...</b>  ","","http:/
               /www.redbooks.ibm.com/abstracts/sg242020.html")

search_result  ROW ("IBM Redbooks | IBM Tivoli Monitoring for Databases Databas
               e <b>...</b> "," <b>...</b> the following three modules: - IBM T
               ivoli Monitoring for DB2 - IBM Tivoli Monitoring<br> for <b>Info
               rmix</b> - IBM Tivoli Monitoring for Oracle In this <b>redbook</
               b>, we show <b>...</b>  ","","http://www.redbooks.ibm.com/abstra
               cts/sg246613.html")

search_result  ROW ("IBM Redbooks | IBM Tivoli Monitoring for Databases: <b>Inf
               ormix</b> An <b>...</b> "," <b>...</b> This Redpaper specificall
               y shows the <b>Informix</b> management using IBM Tivoli Monitori
               ng<br> for Databases: <b>Informix</b> product. . <b>...</b> REDP
               3610. Feedback. <b>redbook</b>@us.ibm.com. <b>...</b>  ","","htt
               p://publib-b.boulder.ibm.com/Redbooks.nsf/RedpaperAbstracts/redp
               3610.html?Open")

search_result  ROW ("IBM Redbooks | A Practical Guide to DB2 UDB Data Replicati
               on V8"," <b>...</b> The objective of this IBM <b>Redbook</b> is
               to provide you with detailed <b>...</b> configure, and<br> imple
               ment replication among the IBM database family - DB2 and <b>Info
               rmix</b>. <b>...</b>  ","","http://publib-b.boulder.ibm.com/Redb
               ooks.nsf/RedbookAbstracts/SG246828.html")

search_result  ROW ("IBM DB2 Developer Domain"," <b>...</b> Up and Running with
                DB2 for Linux This IBM <b>Redbook</b> is an informative guide t
               hat<br> describes how to effectively integrate DB2 UDB with SuSe
                and RedHat Linux <b>...</b>  ","","http://www7b.boulder.ibm.com
               /dmdd/")

search_result  ROW ("www.ruban.de: DB2 Hotline News Archive - Detail View 2002"
               ," <b>...</b> information you can use to install, configure, and
                implement replication among the<br> IBM database family -- DB2
               and <b>Informix</b>. This IBM <b>Redbook</b> is organized so <b>
               ...</b>  ","","http://www.ruban.de/DB2_OS_390/News_Archive/DB2_H
               otline_News_Archive_2002/DB2_Hotline_News_Archive_-_Det/db2_hotl
               ine_news_archive_-_det_69.html")

search_result  ROW ("","  <b>...</b> The storage of these data in <b>Informix</b
               > is complicated, and will be described<br> in a separate docume
               nt. <b>...</b> Deferred. 4.2.10 NCEP (<b>REDBOOK</b>) Graphics.
               <b>...</b>  ","","http://www.nws.noaa.gov/mdl/awips/aifmdocs/sec
               _4_G.htm")

search_result  ROW ("IBM DB2/SAP Newsletter"," <b>...</b> IBM intends to integr
               ate selected <b>Informix</b> technology with DB2 Universal Datab
               ase<br> TM , which will <b>...</b> New <b>Redbook</b>: SAP R/3 o
               n DB2 for OS/390: Database Availability <b>...</b>  ","","http:/
               /www-3.ibm.com/software/data/partners/ae1partners/sap/newslet2q2
               001.html")

search_result  ROW ("IBM Scholars Program: IBM Scholars Data Management Program
               "," <b>...</b> <b>Informix</b> software Through the IBM Scholars
                Data Management Program, <b>Informix</b> products<br> are now a
               vailable to professors, students and researchers to use in the <
               b>...</b>  ","","http://www-3.ibm.com/software/info/university/p
               roducts/data/")

search_result  ROW ("Tivoli Developer Domain"," <b>...</b>  (<b>Redbook</b>) IB
               M Tivoli Monitoring for Databases A special section that covers
               Tivoli<br> software monitoring solutions for DB2, <b>Informix</b
               > and Oracle databases. <b>...</b>  ","","http://www-106.ibm.com
               /developerworks/tivoli/")

10 row(s) retrieved.

> _
```

Figure 9-18 Sample output of the GoogleSearch Java UDR

8. Now we're ready to test the Java UDR to call the Web service. Before you can test the UDR make sure that you're connected to the Internet. Then, for example, start dbaccess to connect to the stores database and execute the GoogleSearch function. Since we're using SQL and SQL doesn't differentiate between lowercase and uppercase letters we just simply type:

   ```
   execute function googlesearch("<search_string>", <start_result>",
   "<language_restriction>", "<topic_restriction>).
   ```

   See Figure 9-18 for some sample results for the Google search string "Informix Redbook"

> **Tip:** If you encounter any "class not found error" messages while executing the GoogleSearch Java UDR you should copy the Google.jar file (as a workaround) also into the $INFORMIXDIR\extend\krakatoa directory and include it in the JVP classpath (or in your JVP classpath file), just after the C:\informix\extend\krakatoa\krakatoa.jar;C:\informix\extend\krakatoa\jdbc.jar and before the other entries. You probably also have to re-start your IDS 9 instance after copying the jar file and modifying the classpath entry.
>
> This behavior occurs because of class loading issues between J/Foundation and AXIS. AXIS uses reflection to load the classes for de/serialization (like Google.DirectoryCategory). Also, the class loader used by AXIS is the system class loader and not the J/Foundation class loader. This could cause problems in the class resolution for classes loaded via install_jar function. It is a good idea to include Google.jar in the JVPCLASSPATH because that way AXIS can find those classes without having to go through the J/Foundation class loader. You can check the Web site related to this redbook for further updates on that topic.

Since we now have the Google search power available on the SQL level, we can do some very cool things. We can first create a data type to match the search results from the GoogleSearch UDR and then create a table to store the search results.

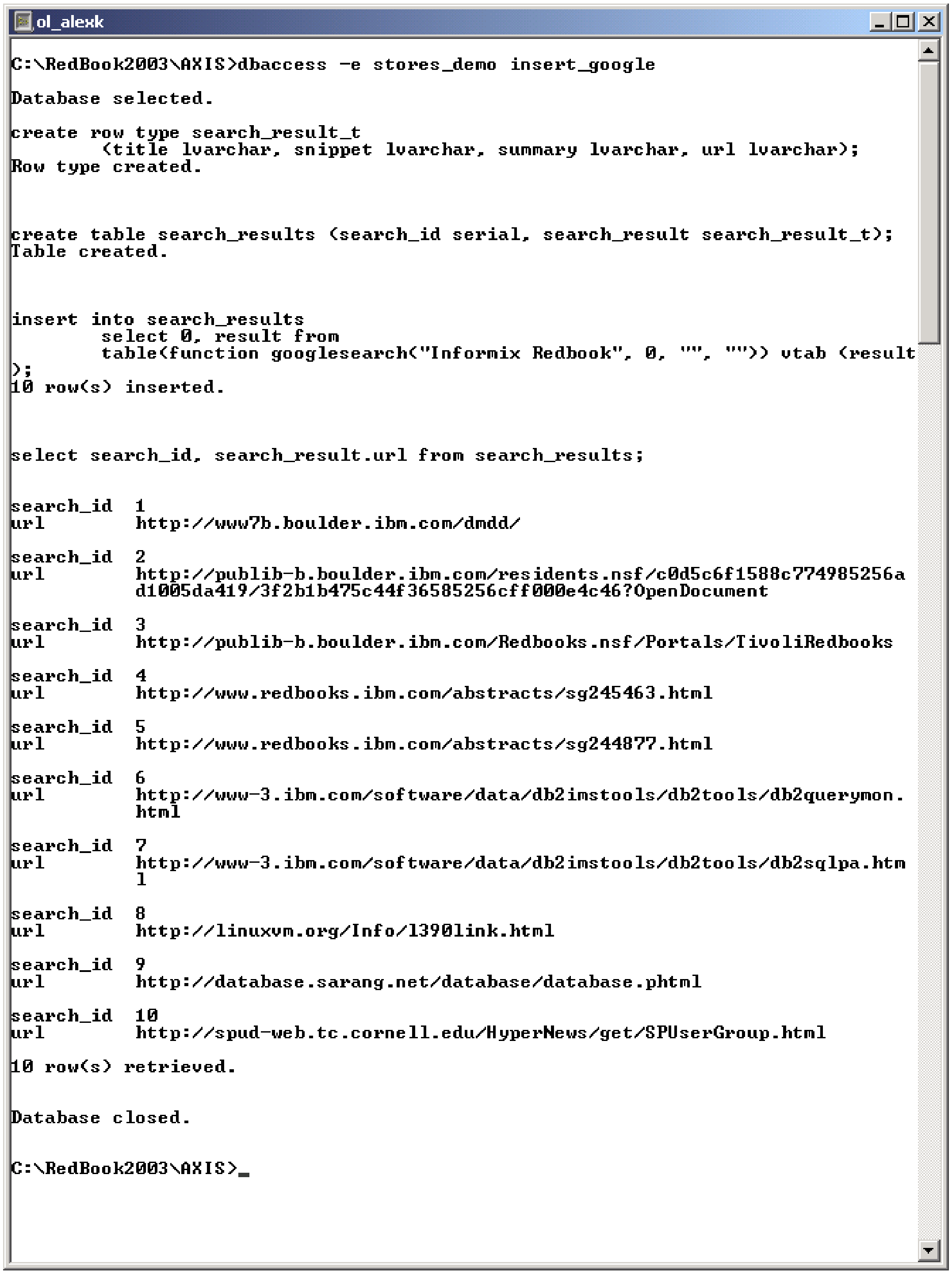
In Example 9-21 you have all the necessary SQL statements to create a ROW type, create a table and then insert the results into that table. As soon we have stored the results, we can then select subsets of our data (Example 9-22).

Example 9-21 SQL script to create a table and insert the results of a Google search (IDS 9 only!)

```
create row type search_result_t
                   (title lvarchar, snippet lvarchar, summary lvarchar,
                         url lvarchar);

create table search_results (search_id serial, search_result
                                                          search_result_t);

insert into search_results
      select 0, result from
            table(function googlesearch("Informix Redbook", 0, "", ""))
                  vtab (result);
```

Example 9-22 Based on the search_results table select only the URL component of each result

```
select search_id, search_result.url from search_results;
```



```
C:\RedBook2003\AXIS>dbaccess -e stores_demo insert_google

Database selected.

create row type search_result_t
        (title lvarchar, snippet lvarchar, summary lvarchar, url lvarchar);
Row type created.



create table search_results (search_id serial, search_result search_result_t);
Table created.



insert into search_results
        select 0, result from
        table(function googlesearch("Informix Redbook", 0, "", "")) vtab (result
);
10 row(s) inserted.



select search_id, search_result.url from search_results;


search_id  1
url        http://www7b.boulder.ibm.com/dmdd/

search_id  2
url        http://publib-b.boulder.ibm.com/residents.nsf/c0d5c6f1588c774985256a
           d1005da419/3f2b1b475c44f36585256cff000e4c46?OpenDocument

search_id  3
url        http://publib-b.boulder.ibm.com/Redbooks.nsf/Portals/TivoliRedbooks

search_id  4
url        http://www.redbooks.ibm.com/abstracts/sg245463.html

search_id  5
url        http://www.redbooks.ibm.com/abstracts/sg244877.html

search_id  6
url        http://www-3.ibm.com/software/data/db2imstools/db2tools/db2querymon.
           html

search_id  7
url        http://www-3.ibm.com/software/data/db2imstools/db2tools/db2sqlpa.htm
           l

search_id  8
url        http://linuxvm.org/Info/l390link.html

search_id  9
url        http://database.sarang.net/database/database.phtml

search_id  10
url        http://spud-web.tc.cornell.edu/HyperNews/get/SPUserGroup.html

10 row(s) retrieved.



Database closed.



C:\RedBook2003\AXIS>_
```

Figure 9-19 You can do neat things with IDS 9.40, Web services and SQL...

Summary

IBM Informix Dynamic Server 9 is a very powerful and flexible platform. It can either act as a Web service provider in combination with the IBM WebSphere product family, or can function as a consumer of Web services, to make them available to SQL-aware applications.

Due to its extensibility features, it is able to adapt to any kind of future application requirements and adjusts easily to the demands placed on it by developers and DBAs.

10

WebSphere Portal Server

In the previous chapters we learned how to use IDS in combination with WebSphere Application Server and WebSphere Studio Application Developer to develop enterprise applications. In order to integrate these applications with existing collaborative applications (such as IBM Lotus® Sametime®) or legacy applications (such as Informix 4GL) and present them to a corporate user, you might want to consider a portal server.

A portal server is also going to help if you have the need to support many different end user devices like PDAs, Notebooks, smart phones, due to the built-in format conversion technology which is XML based. Starting with version 4.2.1, WebSphere Portal Server supports IDS as a repository database.

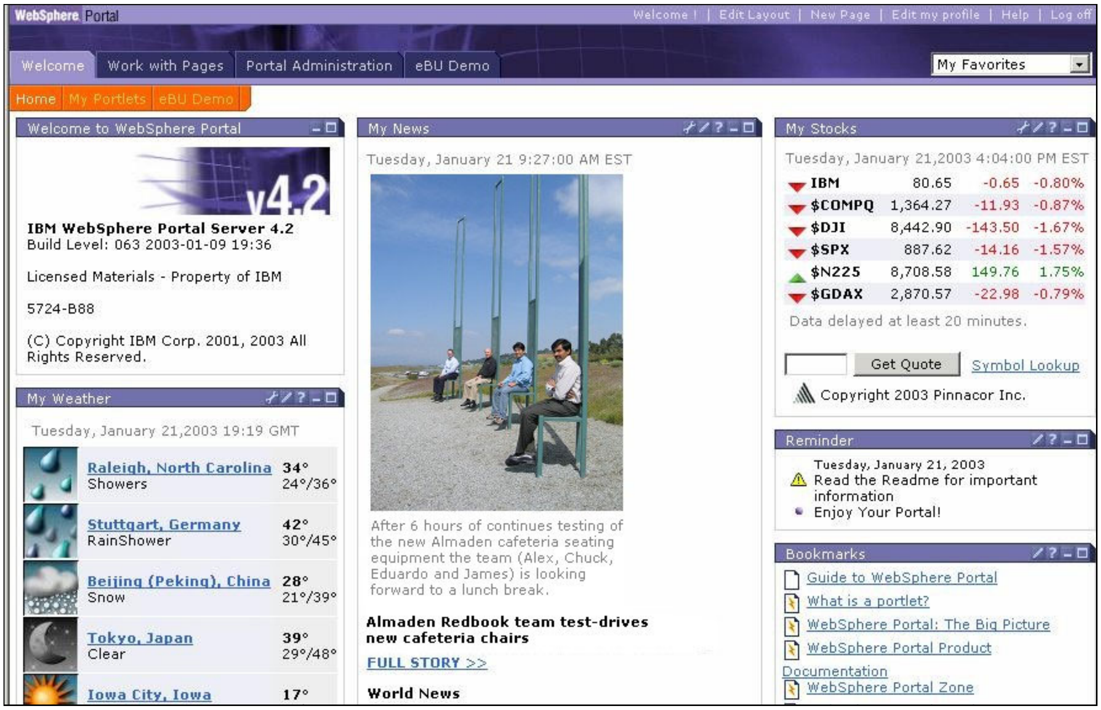
You will find the following topics covered in this chapter:

- Introduction to WebSphere portal server (WPS) and its concepts
- How to configure Informix IDS for WPS

10.1 An introduction to WebSphere Portal

IBM WebSphere Portal for Multi platforms provides a single point of interaction with dynamic information, applications, processes, and people to build successful business-to-employee (B2E), business-to-business (B2B) and business-to-consumer (B2C) portals. WebSphere Portal also supports a wide variety of pervasive devices enabling users to interact with their portal anytime, anywhere, using any device, wired or wireless.

WebSphere Portal consists of three packaged offerings: the Portal Enable offering is the base offering; Portal Extend and Portal Experience both add more functionality. In this chapter, we discuss the three offerings. See Figure 10-1.



IBM WebSphere Portal Server 4.2
Build Level: 063 2003-01-09 19:36

Licensed Materials - Property of IBM

5724-B88

(C) Copyright IBM Corp. 2001, 2003 All Rights Reserved.

Tuesday, January 21,2003 19:19 GMT

Raleigh, North Carolina	Showers	34° 24°/36°
Stuttgart, Germany	RainShower	42° 30°/45°
Beijing (Peking), China	Snow	28° 21°/39°
Tokyo, Japan	Clear	39° 29°/48°
Iowa City, Iowa		17°

Tuesday, January 21 9:27:00 AM EST

After 6 hours of continues testing of the new Almaden cafeteria seating equipment the team (Alex, Chuck, Eduardo and James) is looking forward to a lunch break.

Almaden Redbook team test-drives new cafeteria chairs

FULL STORY >>

World News

Tuesday, January 21,2003 4:04:00 PM EST

IBM	80.65	-0.65	-0.80%
$COMPQ	1,364.27	-11.93	-0.87%
$DJI	8,442.90	-143.50	-1.67%
$SPX	887.62	-14.16	-1.57%
$N225	8,708.58	149.76	1.75%
$GDAX	2,870.57	-22.98	-0.79%

Data delayed at least 20 minutes.

Get Quote Symbol Lookup

Copyright 2003 Pinnacor Inc.

Tuesday, January 21, 2003
Read the Readme for important information
Enjoy Your Portal!

Guide to WebSphere Portal
What is a portlet?
WebSphere Portal: The Big Picture
WebSphere Portal Product Documentation
WebSphere Portal Zone

Figure 10-1 An example WepSphere portal enabled home page

WebSphere Portal Enable

The IBM WebSphere Portal Enable offering allows you to quickly build scalable portals to simplify and speed your access to personalized information and applications. WebSphere Portal Enable provides common services including:

- **Connectivity and integration:** Allows access to enterprise data, external newsfeeds, or even your trading partners' applications.
- **Presentation and administration:** Enables computing desktop customization to match your own work patterns and needs, while providing:
  - Improved productivity with access to enterprise resource planning (ERP), customer relationship management (CRM) and supply chain management (SCM) enterprise applications.
  - Increased security features that include an authentication layer to provide controlled access to the portal, and user information is stored in a Lightweight Directory Access Protocol (LDAP) directory.

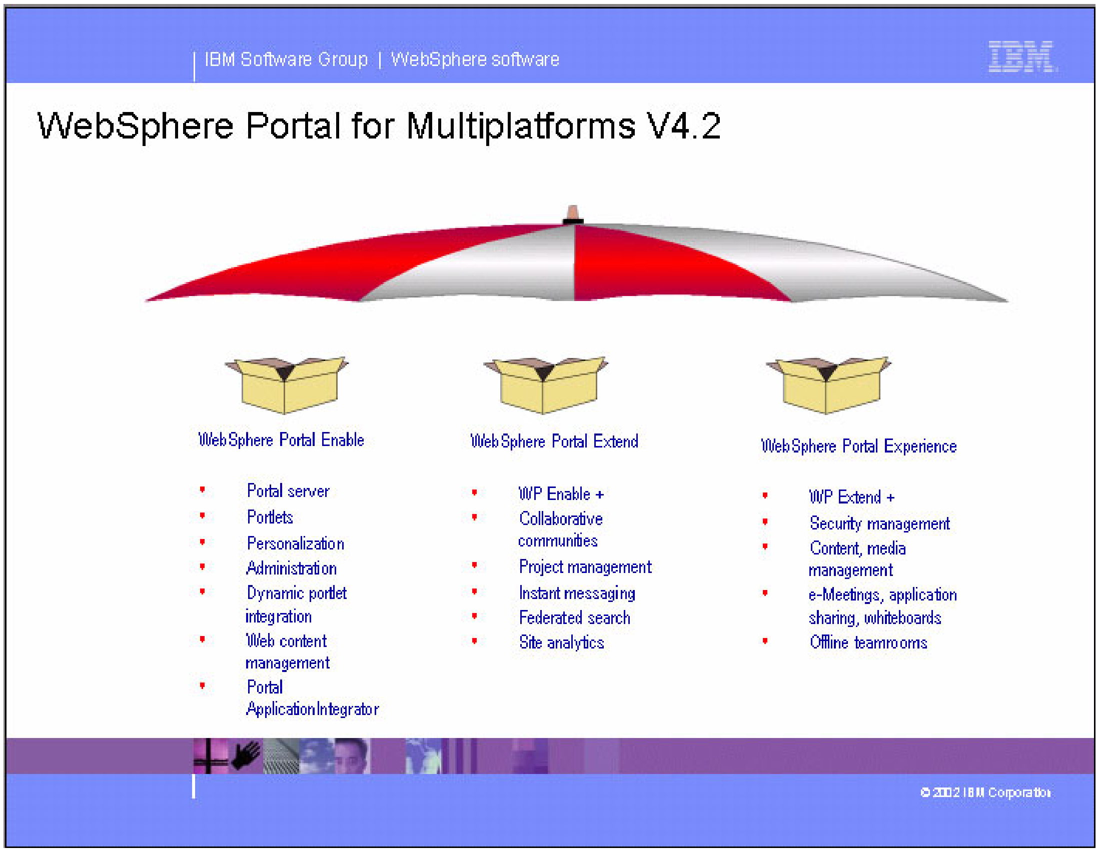
With WebSphere Portal Enable, you can build a Web site that allows users to select which applications they view and how they want to view them. Your site becomes easier to use. Any irrelevant content is filtered out and pertinent content can be quickly located. WebSphere Portal Enable provides two personalization technologies to tailor Web content, including:

- Rules-based filtering to determine which Web content is displayed for a particular user.
- Advanced statistical models and matching techniques to extract visitor behavior and trends, so you can tailor displayed content by individual portlets to different users and groups.

See Figure 10-2. Listed are the WebSphere Portal Enable components:

- WebSphere Portal
- WebSphere Application Server
- WebSphere Personalization
- IBM SecureWay® Directory
- IBM DB2
- IBM Web Content Publisher
- WebSphere Application Developer

Target: Personalized e-business portals that manage content and process transactions.



IBM Software Group | WebSphere software

WebSphere Portal for Multiplatforms V4.2

WebSphere Portal Enable

- Portal server
- Portlets
- Personalization
- Administration
- Dynamic portlet integration
- Web content management
- Portal ApplicationIntegrator

WebSphere Portal Extend

- WP Enable +
- Collaborative communities
- Project management
- Instant messaging
- Federated search
- Site analytics

WebSphere Portal Experience

- WP Extend +
- Security management
- Content, media management
- e-Meetings, application sharing, whiteboards
- Offline teamrooms

© 2002 IBM Corporation

Figure 10-2 The WebSphere Portal 4.2 offerings

WebSphere Portal Extend

Built on the portal framework in the WebSphere Portal Enable offering, the IBM WebSphere Portal Extend offering adds collaborative components and Web analytics coupled with additional tools to access, organize, and share information. Its features include:

- Parallel, distributed, heterogeneous searching capability
- Individual and shared team workspaces with built-in collaborative capabilities
- Collaboration software components
- Web site analyses

Using collaboration technology, WebSphere Portal Extend allows portal users to be more productive because they can collaborate and act on the information they are viewing. Out-of-the-box Web workspaces provide:

- Customizable work environments for individuals, teams or communities.
- The ability to create discussion areas for collaboration about documents stored in document libraries.
- The ability to set up group calendars, assign tasks and communicate through instant messaging.
- Individual collaborative components to make portal and portlet development easy.

WebSphere Portal Extend provides extended search capabilities that allow you to search across an expanded variety of data stores, including relational databases such as IBM DB2 Universal Database™, Oracle, Lotus Notes® and Lotus Domino™ databases, popular Web search engines and text or HTML documents.

WebSphere Portal Extend includes robust Web analysis technology to help you obtain and leverage critical knowledge to optimize your portal. This offering enables you to:

- Make informed decisions about Web initiatives.
- Maximize B2E, B2C and B2B Web site effectiveness for IT, marketing and sales executives.
- Capture, store, measure, report and chart Web site visitor trends and preferences.

WebSphere Portal Extend adds more functionality to WebSphere Portable Enable. Listed are WebSphere Portal Extend components:

- IBM Lotus Collaborative Places
- IBM Lotus Collaborative Components
- IBM Lotus Extended Search
- IBM Tivoli Site Analyzer

Target: B2E and E2E portals requiring robust collaboration with plans to grow on the platform.

10.1.1 WebSphere Portal Experience

In addition to the tools and capabilities contained in IBM WebSphere Portal Extend and IBM WebSphere Portal Enable, IBM WebSphere Portal Experience adds advanced collaboration, content management and security policy management, creating the most comprehensive portal offering in the market. WebSphere Portal Experience allows you to develop, deploy and maintain enterprise portals that provide a first-class experience for employees, trading partners and customers. WebSphere Portal Experience features include:

- Advanced collaboration features for e-meetings, application sharing and whiteboarding enable effective online collaboration as well as the ability to take team rooms offline.

- Data storage for a broad spectrum of digital information including facsimiles, images, PC files, XML, and multimedia.
- Content infrastructure for applications including call centers, high-volume claims processing, and accounts payable.
- Folder management and document workflow.
- Sample Java applications as well as advanced application development tools.
- Security policy management tools for e-business and distributed applications.

WebSphere Portal Experience adds advanced collaboration capabilities and enterprise content management functions, and ensures a more secure portal with security-rich access to information through IBM security management products.

- Advanced collaboration features improve collaboration for mobile users by allowing them to share a screen frame, their desktop, presentations or applications through e-meetings, application sharing and whiteboarding capabilities. Features allow users to create a secure Web workspace instantly, where other users can share ideas and documents and even go off-line.
- Enterprise content management features index, store and distribute digital content quickly and provide the enterprise content management infrastructure to access digital assets created by other business applications. An enterprise-scalable repository allows you to index, store, search and distribute virtually any type of digital content, including HTML and XML Web content, document images, electronic office documents and rich media like digital audio and video.
- Security policy management tools take security to the next level by providing a robust and secure policy management tool that supports e-business and distributed applications. In addition, the secure policy management tool addresses the challenges of escalating security costs, growing complexity and cross-platform security policies.

WebSphere Portal Experience adds more functionality to WebSphere Portable Enable and WebSphere Portal Extend. Listed are WebSphere Portal Experience components:

- IBM Content Manager
- IBM Tivoli Access Manager
- IBM Lotus Sametime
- IBM Lotus QuickPlace

Target: Comprehensive e-business portals requiring advanced security, content management and collaboration capabilities.

10.1.2 Industry impact and acceptance

Industry research has indicated that IBM has significantly improved its product offering of WebSphere Portal requiring much less services for deployment. The significant achievements that have been made are listed:

- IBM has constructed an Enterprise Portal solution which is very impressive; its technology has a wealth of features that are not available, at least together, in competitive products.
- The ease with which WebSphere Portal may be managed and its comprehensive capabilities to delegate administration of sections of the portal environment are impressive. This ensures that flexible, secure and manageable portal environments can be created in a cost effective, responsive manner.
- WebSphere Portal is a solution for business-to-business (B2B) and business-to-consumer (B2C) environments and it provides application integration for all enterprise Web-based environments.

- IBM provides a range of product offerings, ranging from entry-level portals to those with a true enterprise-wide scope making WebSphere Portal available to business of all sizes.
- The fact that WebSphere Application Server forms the foundation of this solution means that issues such as security, scalability and reliability should not be an issue. WebSphere Portal is built on top of WebSphere Application Server Version 4 technology ensuring compliance with J2EE standards.

10.2 WebSphere Portal architecture

The WebSphere Portal platform is positioned to enhance the WebSphere family of products, providing tooling for aggregating and personalizing Web-based content and making that content available via multiple devices. WebSphere Portal takes advantage of the strong platform provided by WebSphere Applications Server. See Figure 10-3.

WebSphere Portal finds its roots in Apache Jetspeed. Jetspeed is an Open Source implementation of an Enterprise Information Portal, using Java and XML. Jetspeed was created to deliver an Open Source Portal that individuals or companies could use and contribute to in an Open (Source) manner.

Soon after creation, it became apparent that Jetspeed was going to become an "engine" for Web applications. That, however, was far beyond the scope of the original project. Around that time, there were many discussions on the mailing list that spawned the Turbine project based on technology donated by Jon Stevens/Clear Ink. Turbine is now the Web Application framework that Jetspeed shares with many other Web applications.

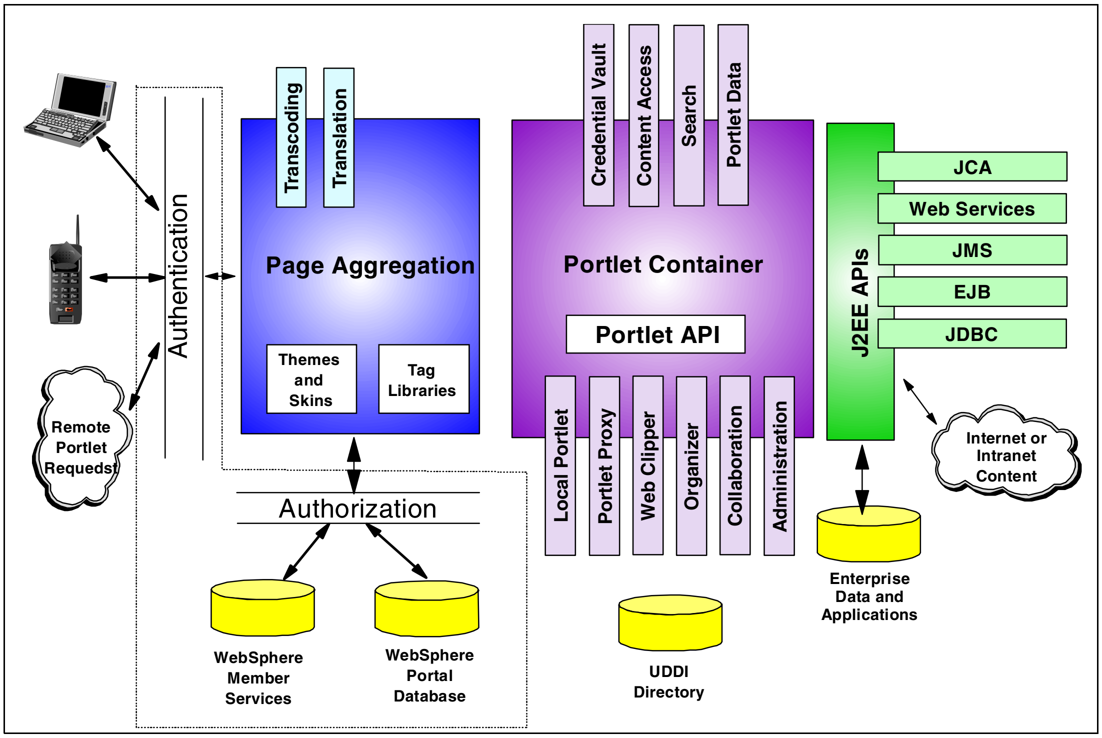


Figure 10-3 WebSphere Portal Architecture

Building on the Jetspeed implementation, WebSphere Portal provides an architecture for building and running portal applications. The overall WebSphere Portal Architecture can be seen in Figure 10-3. WebSphere Portal provides services for Authentication and Authorization though the WebSphere Member Services. The core of WebSphere Portal architecture is composed of the Presentation Services, the portal infrastructure, and the portal services.

Presentation services

WebSphere Portal presentation services provide customized and personalized pages for users though aggregation. Page content is aggregated from a variety of sources via content and applications. The portal presentation framework simplifies the development and maintenance of the portal by defining the page structure independent the portlet definition. Portlets can be changed without impact to the overall portal page structure.

The Portal engine

WebSphere Portal provides a pure Java engine whose main responsibility is to aggregate content from different sources and serve the aggregated content to multiple devices. The Portal engine also provides a framework that allows the presentation layer of the portal to be decoupled from the portlet implementation details. This allows the portlets to be maintained as discrete components. Figure 10-4 shows the WebSphere Portal Engine Components.

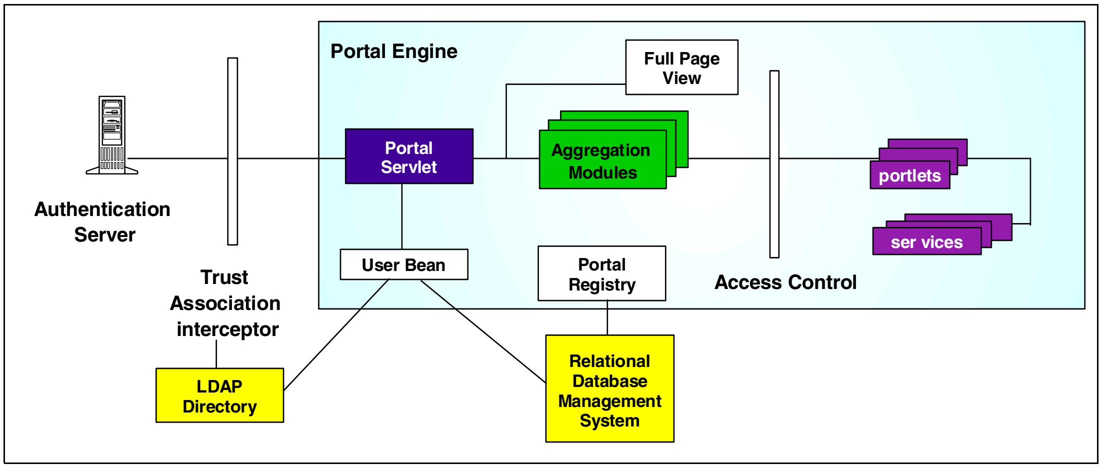


Figure 10-4 The WebSphere portal engine

The Authentication Server is a third party authentication proxy server that sits in front of the Portal engine. Access to portlets is controlled by checking access rights during page aggregation, page customization, and other access points.

The Portal Servlet is the main component of the portal engine. The portal servlet handles the requests made to the portal. The portal requests are handled in two phases. The first phase allows portals to send event messages between themselves. In the second phase, the appropriate Aggregation Module for the requesting device renders the overall portal page by collecting information from all the portlets on the page and adding standard decorations such as title bars, edit buttons, etc.

Portlet container

Portal Services are components WebSphere Portal uses to extend the portal functionality. Key functionality is provided with WebSphere Portal for personalization, search, content management, site analysis, enterprise application integration collaboration and Web services. Portlets can access these services via their container.

Portal infrastructure

The WebSphere Portal infrastructure is the framework that provides the internal features of the portal. Functionality such as user and group management via self registration, as well as portal administration, are provided by the Portal infrastructure.

User and group management

The WebSphere Portal infrastructure provides facilities to allow user self management along with enterprise integration with user directories such as LDAP or database structures.

Security services

As WebSphere Portal runs within the WebSphere Application Server platform, it makes use of the standard Java Security APIs to provide authentication. The WebSphere Portal is configured so that incoming requests pass through an authentication component such as WebSphere Application Server, WebSEAL (a component of SecureWay) or other proxy servers. A user's authorization for a particular resource such as page or a portlet is handled by the portal engine.

User Beans are provided to allow programmatic access to the User information for use within portlets.

Page transformation

WebSphere Transcoding Technology is integrated with WebSphere Portal to transform the portal markup produced by WebSphere Portal to markup for additional devices such as mobile phones and PDAs.

Portal services

Portal services are built-in features the WebSphere Portal provides to extend and enhance the full portal solution. These services are provided via the Portlet container as shown in Figure 10-3 on page 258. Among the services are:

- **Personalization:** The IBM WebSphere Personalization functionality enables advanced personalization capabilities. Base customization, such as choosing which portlets are desired on a page, is accomplished by the user via administration functionality. Advanced personalization via rules engines, user preferences and profiles is accomplished by the provided personalization services.
- **Content management:** WebSphere Portal provides services to facilitate connections to content management sources. Built-in support is provided for several common content types such a as Rich Site Summary (RSS), News Markup Language (NewsML) and Open Content Syndication (OCS) along with most XML and Web browser markup.
- **Search:** WebSphere Portal offers a simple search service. The Portal Search capability enables search across distributed HTML and text data sources. The search can crawl a Web site and is configured so as to force it to follow several layers in a site or to extend beyond several links in a site. Furthermore, IBM Extended Search and Enterprise Information Portal can be fully incorporated into the portal environment. These search engines are industrial-strength tools that provide federated searches across numerous data sources.
- **Site analysis:** You can take advantage of the underlying WebSphere Application Server technology and Site Analyzer to provide information about Web site visitor trends, usage and content. This detailed information can then be used to improve the overall effectiveness of the site.
- **Collaboration:** Collaboration services are provided by WebSphere Portal through a set of pre-defined portlets. These portlets allow for team-room function, chat, e-mail, calendering and many other collaborative technologies.

- **Web services:** WebSphere Portal provides services for exposing and integrating portlets as remote portlets hosted on another portal platform via Web Services technology. The entire process of packaging and responding to a SOAP request is hidden from the developer and the administrator.

10.2.1 WebSphere Portal tooling

WebSphere Portal and WebSphere Portal Toolkit, along with their prerequisite products, provide the basic tooling for developing and deploying portals and their associated portlets.

WebSphere Portal

WebSphere Portal contains built-in support for portlet deployment, configuration, administration and communication between portlets.

WebSphere Portal provides the framework for building and deploying portals and the portal components, portlets. Portlet content is aggregated by the WebSphere Portal to provide the desired portal implementation.

WebSphere Portal makes use of the WebSphere Application Server technology to provide a portal platform.

WebSphere Portal Toolkit

The WebSphere Portal Toolkit is provided with WebSphere Portal and provides an environment for developing portal using WebSphere Portal. The WebSphere Portal Toolkit is a plug-in for WebSphere Studio Application Developer (WSAD) or WebSphere Studio Site Developer (WSSD) which adds the portal development environment.

The WebSphere Portal Toolkit provides the ability to quickly create complete, MVC-compliant portlet applications. It also provides intuitive editors for working with the deployment descriptors required by your portlet applications. Furthermore, it allows you to dynamically debug your portlet applications.

10.3 WebSphere Portal

WebSphere Portal takes advantage of the WebSphere Application Server base, making use of its J2EE services. WebSphere Portal itself installs as an Enterprise application in WebSphere Application Server.

10.3.1 Portal concepts

In this section we provide definitions of some basic portal concepts.

Portlet

A portlet is an application that displays page content.

Portlet application

Portlet applications are collections of related portlets and resources that are packaged together. All portlets packaged together share the same context which contains all resources such as images, properties files and classes. Important also is the fact that portlets within a portlet application can exchange messages.

Page

A portal *page* displays content. A page can contain one or more portlets. For example, a World Market page might contain two portlets that displays stock tickers for popular stock exchanges and a third portlet that displays the current exchange rates for world currencies. To view a page in the portal, you select its place from the place selector and then click the page within the place.

Place or page group

A *place* is a collection of portal pages. The portal administrator can create places, determine which portal pages are in the each place, and give the appropriate users authority to access the place and pages.

Layout

The page *layout* defines the number of content areas within the page and the portlets displayed within each content area. In many cases, the portal administrator defines the page layout. The administrator can permit specified users or user groups to change the page layout to reflect individual preferences. If you have authority to change a page, use the Layout page in the Work with Pages place to alter the page layout.

Permissions

Each portal page is subdivided into one or more content areas. Each content area can contain one or more portlets. The portal administrator or a user who has authority to manage a page can control whether others who have authority to edit the page can move, edit or delete the content areas and the portlets on the page. *Permissions* is the term for controlling those settings. If you have authority to make changes to a portal page, use the Permissions page in Work with Pages place to set the permissions for the page.

Themes

Themes represent the overall look and feel of the portal, including colors, images and fonts. There are several default themes provided with the standard installation of WebSphere Portal. Each place in the portal may have a different theme associated with it, thereby creating the appearance of virtual portals.

Skins

The term *skin* refers to the visual appearance of the area surrounding an individual portlet. Each portlet can have its own skin. The skins that are available for use with a portlet are defined by the portal theme that is associated with the place. The portal administrator or the designer determines the theme for places and the available skins for the theme. The administrator can permit specified users to change the skins to reflect individual preferences. If you have authority to make changes to a portal page, use the Skins page in Work with Pages to set the skins for portlets.

10.3.2 Portlets

The base building blocks of a Portal are the portlets. Portlets are complete applications following the Model-View-Controller design pattern. Portlets are developed, deployed, managed and displayed independent of all other portlets.

Portlets may have multiple states and view modes along with event and messaging capabilities. Based on the J2EE container model, portlets run inside the Portlet Container of WebSphere Portal analogous to the way servlets run inside the Servlet Container of WebSphere Application Server. Portlets are a special subclass of HTTPServlet that includes properties and functionality that allows them to run inside the Portlet Container.

Though portlets actually run as servlets under the WebSphere Application Server, they cannot send redirects or errors to the browser directly, forward requests or write arbitrary markup to the output stream. All communication back to the end user from a portlet is done via the aggregation modules.

To understand the portlet model used by WebSphere Portal, let us take a step back and examine the Flyweight pattern. This pattern is used by WebSphere Portal as the design pattern for the portlet model.

The Flyweight pattern

The Flyweight pattern was originally presented by the GofF or Gang of Four (Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides) in E.Gamma, et al*., Elements of Reusable Object-Oriented Software*, Addison Wesley, 1995.

Flyweight is a structural pattern used to support a large number of small objects efficiently. Several instances of an object may share some properties. Flyweight factors these common properties into a single object, thus saving considerable space and time otherwise consumed by the creation and maintenance of duplicate instances. Key to the Flyweight Design Pattern is the fact that the objects share some information. It is then possible to greatly reduce the overhead problem and make the presence of so many objects possible.

The flyweight object is a shared object that can be used in multiple contexts at the same time; the object functions independently in each context.

The state shared by the objects falls into two categories, intrinsic and extrinsic.

Intrinsic state State stored in the object and independent of object's context. Thus the information is sharable across the objects. The more stateless and intrinsic information shared between objects in the flyweight, the better. This allows for greater savings in memory, since less context information needs to be passed around.

Extrinsic state State that depends on a single request varies with the objects context and therefore cannot be shared. This information must be stateless and determined by context, having no stored values, but values that can be calculated on the spot. Client Objects are responsible for passing the extrinsic state to the object when the object needs it.

This separation into extrinsic and intrinsic information allows great numbers of similar objects to exist, differing only in the context in which they exist.

The different components involved in the Flyweight Pattern are the Flyweight, the ConcreteFlyweight, the UnsharedConcreteFlyweight, the FlyweightFactory and the Client.

Flyweight The shared object with intrinsic state. The flyweight declares an interface through which flyweights can receive and act on intrinsic data.

ConcreteFlyweight Implements the Flyweight interface and adds storage for the intrinsic state.

UnsharedConcreteFlyweight The flyweight interface enables sharing but does not enforce it. Not all flyweights are shared. It is common for UnsharedConcreteFlyweight objects to have ConcreteFlyweight objects as children at some level in the hierarchy.

FlyweightFactory	Serves to dispense particular flyweights that are requested. When a Flyweight with certain properties is requested, it checks to see if one already exists, and if so, returns that flyweight. If the requested flyweight does not exist, it creates the requisite flyweight, stores and returns it.
Client	When creating an object, a client must assign a flyweight to it, so it asks the FlyweightFactory for a particular flyweight, receives that flyweight, and creates a reference to it in the object it is creating.

The parameterization of portlets is based on the flyweight pattern, the Portlet Container being the Flyweight Factory.

Portlets

Portlets are invoked by the portlet container. Every portlet has to inherit from the abstract org.apache.jetspeed.portlet.Portlet class, either by deriving directly from it, or by using one of the abstract portlet implementations.

A portlet is a small Java program that runs within a portlet container. Portlets receive and respond to requests from the portlet container. There is ever only one portlet object instance per portlet configuration in the Web deployment descriptor. There may be many PortletSettings objects parameterizing the same portlet object according to the Flyweight pattern, provided on a per-request basis.

When the portal administrator deploys a new portlet or copies an existing one, PortletSettings are created. A Portlet parameterized by its PortletSettings is referred to as a *concrete portlet.* The settings of concrete portlets may change at runtime since administrators modify the portlet settings by using the configuration mode of the portlet. The PortletSettings initially contain the elements defined in the deployment descriptor and are changeable by the portlet administrator. See Figure 10-5.

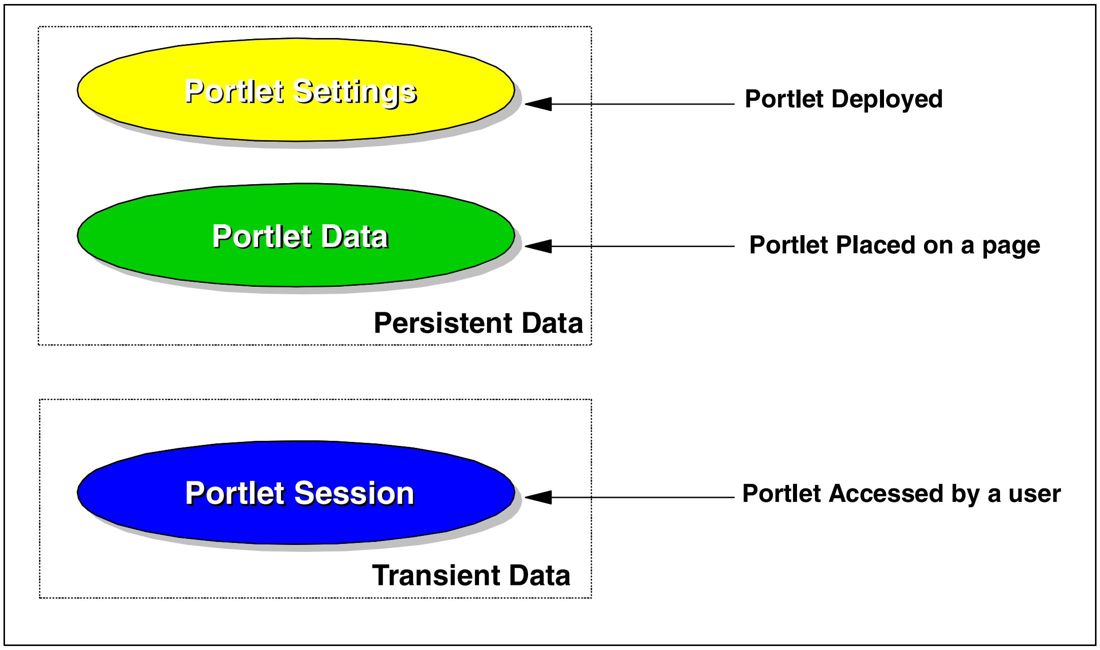


Figure 10-5 Portlet Parameterization objects

Additionally, users can have personal views of concrete portlets. Therefore, the transient PortletSession and persistent concrete PortletData carries vital information for the portlet to create a personalized user experience.

When a portlet is added to a page, PortletData is created to parameterize the portlet. The PortletData can only be accessed by the portlet itself, for example when changing a list of desired stocks to watch in a stock portlet. A concrete portlet in conjunction with portlet data creates a *Concrete Portlet Instance*. PortletData scope depends on the scope of the page. If the administrator places the portlet on a page, the portlet data contains data stored for the group of users associated with the page. If a user puts the portlet on the age, the portlet data contains data for that user.

Finally, when the user initially accesses a portlet, a PortletSession is created. The portlet session stores transient data associated with an individual use of the portlet. The concrete portlet instance parameterized by the PortletSession is referred to as the *User Portlet Instance*. See Figure 10-6.

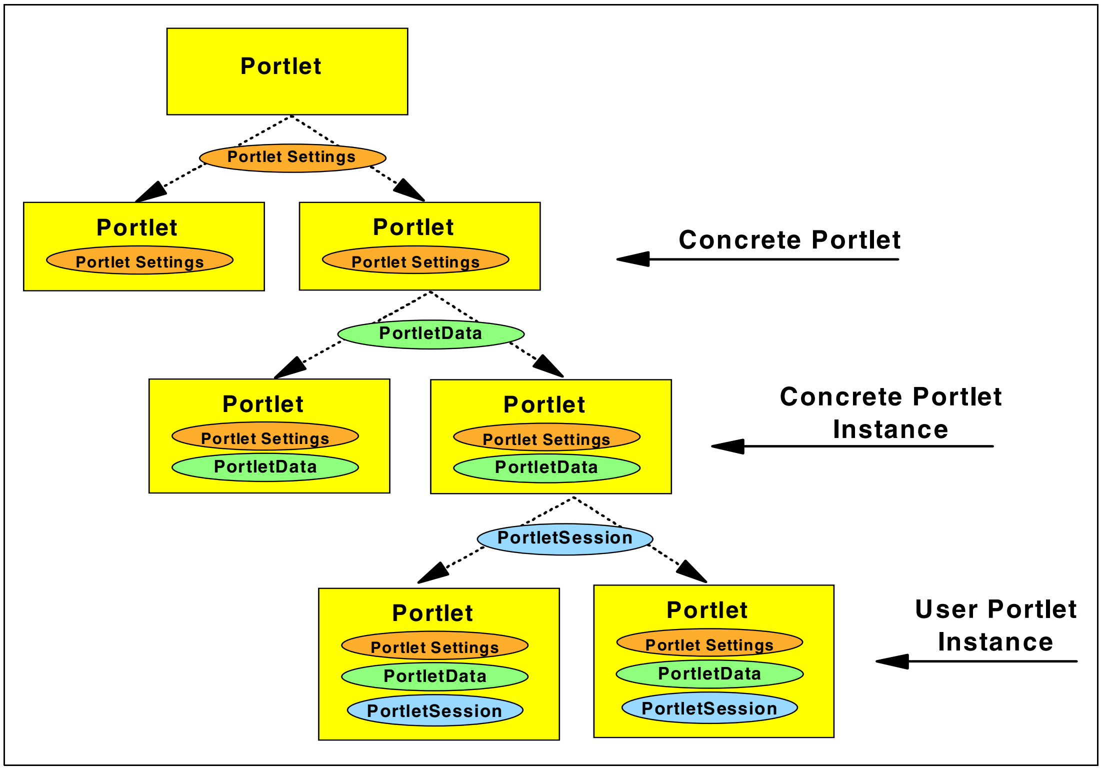


Figure 10-6 The Portlet parameterization

Portlet modes

Portlet Modes are a facet of the Portal display model. Modes allow the portlet to display a different “face” depending on its usage. There are four modes:

View Initial face of the portlet when created. The portlet normally functions with this face.

Help If the portlet supports the help mode, a help page will be displayed for the user.

Edit This mode allows the user to configure the portlet for their personal use, for example, specifying a city for a localized weather forecast.

Configure If provided, this mode displays a face that allows the portal administrator to configure the portlet for a group of users or a single user.

Portlet states

Portlet states determine how the portlet is displayed in the portal. The state of the portlet is stored in the PortletWindow.State object and can be queried for optimizing processing based on state. The three states of a portlet are:

Normal Portlet is displayed in its initial state as defined when it was installed.

Maximized The portlet view is maximized and takes over the entire body of the portal replacing all the other portal views.

Minimized Only the portlet title bar is visible inside the portlet page.

Portlets and the model-view-controller (MVC) design pattern

Because portlets must be capable of supporting multiple views for multiple devices, the key design pattern used for portlets is the *model-view-controller* (MVC) design pattern. This design pattern contains three entities:

- **The model:** This is the data source to be retrieved for the portlet. Model data for a portlet is typically retrieved from an external data source and loaded into Java display beans, or arrives formatted in an XML document.
- **The view or views:** This is the output mechanism used to display the data of the portlet. Display views are typically implemented as either JSPs, more typically used when the data model is implemented in Java beans, or XSLT style sheets when the incoming data is formatted in an XML document.
- **The controller:** This joins the selected view to the data and conducts the operation of the portlet. The controller selects the view for display based on the target device or browser, and then passes the data model to the view. The view extracts the specific display data, formats the data for the browser, and renders its output to the browser as part of the portal aggregation of portlet outputs.

For portlet development, the MVC pattern has the following characteristics:

- The portlet is only responsible for calling the right controller, depending on the markup supported by the client.
- Connectors are responsible for accessing content sources. Typically, there is one connector per content source type, for example, one connector for POP3 access and one for file-based cache.
- Models represent the content as retrieved through the connector. A model is independent of the presentation.
- Controllers can be used to provide markup-specific content (HTML, cHTML, or WML).

In the MVC structure, there is a distinct separation of data from presentation along with a controller component for managing the interaction between the data (model) and the presentation or view. The controller knows the environment in which the application is invoked, gathers information from the data object to be displayed, and then applies the appropriate view to render the data using the markup language appropriate for the current device.

WebSphere Portal Runtime: the portlet container

WebSphere Portal is a J2EE application based on the servlet technology. In fact, portlets inherit from HTTP Servlet in the Java hierarchy, providing the servlet functionality. The WebSphere Portal portlet container is not, however, a standalone container as is the servlet container. The portlet container is a thin layer implemented on top of the servlet container designed to reuse the functionality provided by the servlet container.

The Portlet API provides the standard interfaces for accessing the services provided by the portlet container. As previously mentioned, the Portlet Container is implemented on top of the servlet container and thus the Portal API is very similar to the servlet API.

10.3.3 Portlet lifecycle

Much like the Servlet Container, the Portlet Container manages the portlet lifecycle along with providing services to the portlets running in the container.

The portlet container loads and instantiates the portlet class. This can happen during startup of the portal server or later, but no later then when the first request to the portlet has to be serviced. Also, if a portlet is taken out of service temporarily, for example while administrating it, the portlet container may finish the lifecycle before taking the portlet out of service. When the administration is done, the portlet will be newly initialized.

During the portlet lifecycle, the portlet container invokes the following methods on the Portlet class (subclass of a the Portlet Adapter class) on behalf of user requests, as shown in Figure 10-7.

- init()
- initConcrete()
- login()
- service()
  - doView()
  - doEdit()
  - doHelp()
  - doConfigure()
- logout()
- destroyConcrete()
- destroy()

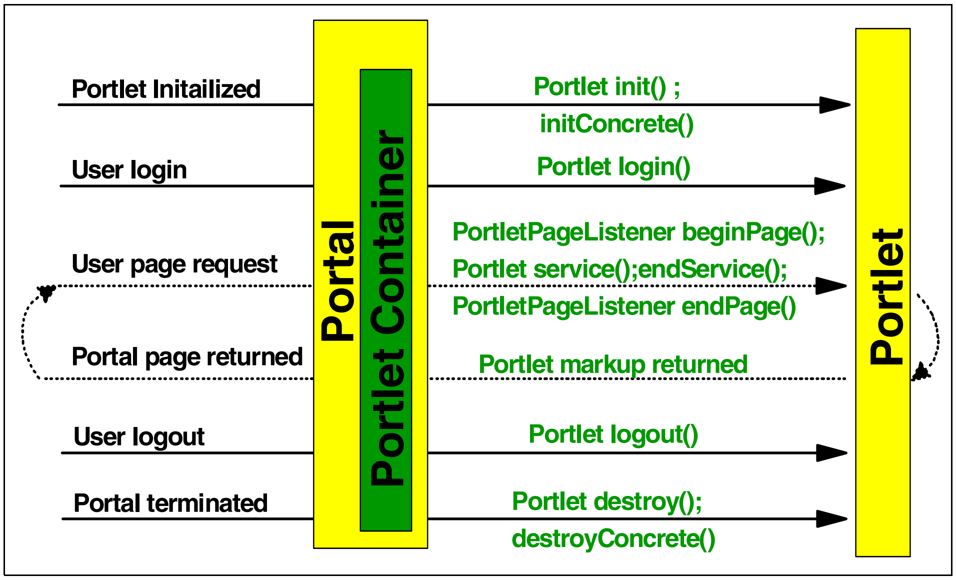


Figure 10-7 Portlet Lifecycle

The portlet container calls the following methods of the abstract portlet during the portlet's life cycle:

- **init():** The portlet is constructed after portal initialization and initialized with the init() method. The portal always instantiates only a single instance of the portlet, and this instance is shared among all users, exactly the same way a servlet is shared among all users of an application server.
- **initConcrete():** After constructing the portlet and before the portlet is accessed for the first time, the portlet is initialized with the PortletSettings. This is known as the concrete portlet.
- **service():** The portal calls the service() method when the portlet is required to render its content. During the lifecycle of the portlet, the service() method is typically called many times. For each portlet on the page, the service() method is not called in a guaranteed order and may even be called in a different order for each request.
- **destroyConcrete():** The concrete portlet is taken out of service with the destroyConcrete() method. This can happen when an administrator deletes a concrete portlet during runtime on the portal server.
- **destroy():** When the portal is terminating, portlets are taken out of service, then destroyed with the destroy() method. Finally, the portlet is garbage collected and finalized.

10.3.4 Portlet events and messaging

Many portals today display static content in independent windows. The ability for portlets to interact within a portal is key to giving a portal a “live” feeling. In “live” portals, quite often the user is presented with one portlet on a page that presents a choice of data, a list of stocks for example, and choosing from the list causes another portlet to be updated with the details of the choice. This type of list detail processing via multiple portlets is done with portlet events and messaging.

This same type of process could be accomplished using a single portlet but consider the example of a stock list, stock details and news associated with the stock. Giving the user this function via three portlets allows the user to customize the portal experience by choosing which information about the chosen stock is displayed by simply adding the associated portlet to the page.

In portlet messaging, one portlet typically detects a condition and formats a message as a result of that condition, then sends the message to the receiver. The receiving portlet receives the message from the event handler and processes the message as you would expect. Portlets can both send and receive messages.

Portlets communicate using portlet actions and portlet messages. For example, an 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 to the transaction detail portlet.

There are some basic rules to portlet messaging:

- Portlets in different applications can only communicate through default portlet message objects. Default portlet message objects can only carry strings.
- In order for portlets to communicate through custom messages, they must be part of the same portlet application. WebSphere Portal uses a unique class loader for each portlet application to provide security between applications. The message is typically a custom Java object unique to the application. Since messaging portlets must share this message object, they must share the same class loader and therefore they must be part of the same portlet application.

- For performance reasons, portlets that communicate through messaging must reside on the same page. Since only one page is displayed at a time, there is little need to send messages to portlets not currently displayed.

Portlet events contain information about an event to which a portlet might need to respond. For example, when a user clicks a link or button, this generates an action event. To receive notification of the event, the portlet must have the appropriate event listener implemented within the portlet class.

Action events: Generated when an HTTP request is received by the portlet container that is associated with an action, such as when the user clicks a link.

Message events: Generated when another portlet within the portlet application sends a message.

Window events: Generated when the user changes the state of the portlet window.

The portlet container delivers all events to the respective event listeners (and therefore the portlets) before generating any content to be returned to the portal page. Should a listener, while processing the event, find that another event needs to be generated, that event will be queued by the portlet container and delivered at a time point determined by the portlet container. It is only guaranteed that it will be delivered and that this will happen before the content generation phase. There is no guarantee for event ordering.

Once the content generation phase has started, no further events will be delivered. For example, messages cannot be sent from within the service, doView or other content generation methods. Attempts to send a message during the content generation phase will result in an org.apache.jetspeed. portlet.AccessDeniedException.

The event listener is implemented directly in the portlet class. The listener can access the PortletRequest.

It is important to understand the underlying event handling and message processing to ensure delivery of all send messages. The portal event handling and message processing sees four steps executed in the following order:

1. **Processing all action events:** The user makes a request of the portal, the portal receives the request and decodes the action URI sent by the client and propagates an action event to the appropriate portlet. The receiving portlet's action listener is called to process an action event. An appropriate time to send messages to other portlets is during the processing of the action event.
2. **Processing all message events:** If a message is sent to a portlet, the portlet's message listener is called to process the message. Since portlets can send multiple messages and send messages as a result of receiving a message, this process continues until there are no more messaging events pending. Cyclical messaging is prevented by the WebSphere Portal architecture.
3. **Processing all window events:** Sizing operations such as maximize, minimize and restore, along with the portlet's ability to request a specific size, causes multiple window events to be sent to all portlets affected by the sizing activity. This processing of window events continues until there are no more window events pending.
4. **Portlet rendering process:** Upon completing the event processing in the order specified above, the portal aggregator begins calling each container on the page being displayed, causing its contents to be rendered. The rendering process is explored in detail in “Page aggregation” on page 270. When aggregation is complete, the page is returned to the user.

Important: It is important to note that events are not processed in the last step of the process, page rendering. If a message is sent by a portlet during rendering, the message will not be delivered or processed. This is a result of the fact that the event queue is help in the portlet request and at the time of rendering, the portlet request is no longer available. Therefore, if portlet interaction is desired, portlet messages must be sent during the first three steps of the event and aggregation process.

10.3.5 Page aggregation

Portals allow users to choose sets of portlets they would like to work with and provides a framework for displaying those portlets in a consistent fashion.

A defined set of applications, which should be presented in a common environment are referred to as a *page*.

Page aggregation is the process that collects information about the user's choices, the device being used and the selected portlets, then takes that information and combines it to create a display that is appropriate for the device.

The aggregation process involves three basic steps:

1. Collecting user information
2. Selecting the active applications
3. Aggregating the output

Once the active page is determined, the layout of this page is used to aggregate the content of the defined applications, arrange the output and integrate everything into a complete page. Basic Portal Page Layout can be seen in Figure 10-8.

Rendering of page components is done using JSPs, images, style sheets, and other resources. These resources are located in the file system in a path-naming convention that the portal server uses to locate the correct resources for the client. WebSphere Portal provides dynamic aggregation of pages from page descriptors held in the portal database.

Collecting user information

During the collection of user information, the following information is collected:

User The user is authenticated at login and the user identification is available throughout the session.

Client The user's device is determined by information contained in the request header. Once determined, this information is also stored in the session.

Markup The markup is associated with the device category. There are currently three markups defined, HTML, cHTML and WML. New markup scan be added via the Markup Manager Portlet.

Markup version The version for the supported markup. For example, `ie5` for the Internet Explorer family of browsers, `ns` for the Netscape family of browsers.

Language The portal determines the language to be displayed via this algorithm:

If the user is logged in, the portal user interface is displayed in the preferred language of the user.

If no preferred language is set, the portal UI is displayed in the language set by the client browser if available.

If no browser language is available, the portal UI is displayed in the default language set for the portal.

Portlets not supporting any of the above scenarios display their UI in the portlet’s default language.

Page/page groups The access control list determines which pages and page groups a user has access to.

Theme The name of the active theme is taken from the currently active page group.

Screen Depending on the interactions of the user with the portal, different screens are presented. The screen holds the output of the portlets on a page.

Selecting the active applications

During this phase of aggregation, the portal determines the active applications or portlets to be displayed. When the portal receives a request, it determines the active place and the active page for the current user. Aggregation then continues with the rendering of the page.

Aggregating the output

Once the active page is determined, the portal uses the layout of the page to aggregate the content of the defined applications, to place the output and build the complete page. A page contains components such as row or column containers that contain other components or portlets. Figure 10-8 shows the layout of a portal page.

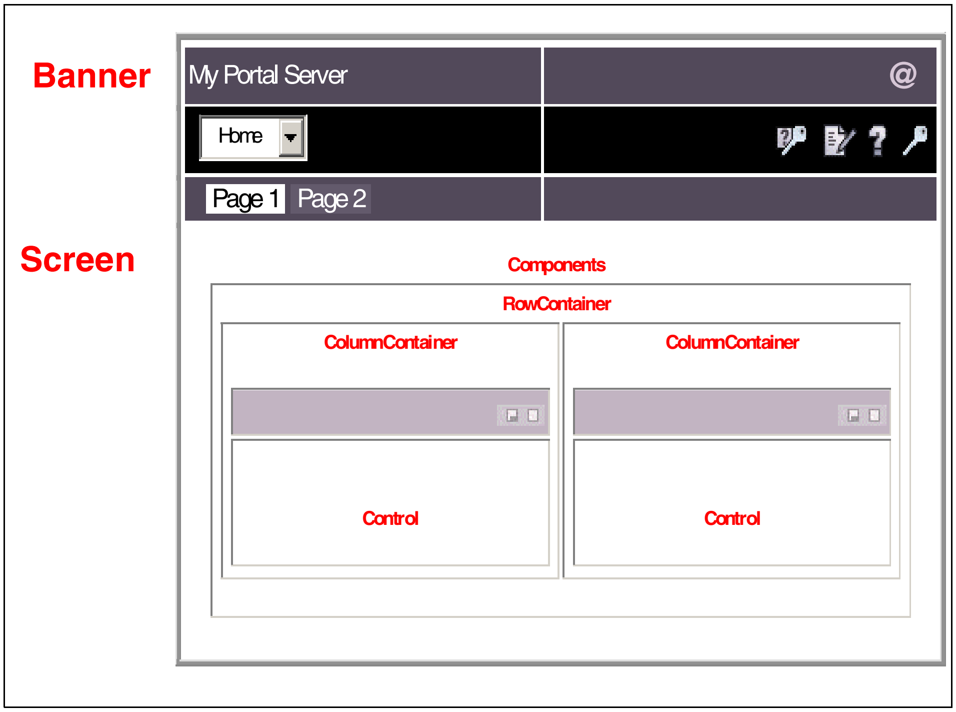


Figure 10-8 Portal page layout

A portal page is made up of the following elements:

Portal window The content inside the displayed window. It is made up of the banner and the portal page.

Banner The top area of the window that holds the company information, the greeting, a page selection box, tabs to select the current page in the page group being displayed and some additional controls for interacting with the portal such as logging in, logging out and help.

Screen Holds the output of the portlets on the currently selected page. The layout is determined by its row and column containers.

Row A container inside a page that allows portlets to be arranged in a horizontal format.

Column A container inside a page that allows portlets to be arranged in a vertical format.

Control The frame around the portlet is constructed by the frame. It builds the bar above the portlet output including buttons to control the state and view of the portlet.

Themes and skins

Window and component layouts can be controlled by themes and skins. Themes refer to the window templates. Themes represent the look and feel of the portal, including fonts and colors, and is also used to render to portal banner. Skins refer to the component templates. Skins use the theme name to select the graphics that match the theme colors.

Templates

Aggregation uses the concept of templates to perform window, screen and component layout. When a corresponding part needs to be rendered, a template loader will load the requested template. If the requested template can't be found, the default template will be used. A template consists of the template class that controls the rendering, the localization and the launch of the template JSP. The template JSP performs the actual rendering. There are three types of templates:

- **Window templates:** The Window template is responsible for the layout of the parts of the banner area and the placement of the screen. You can change, for example, the navigation tab location via the window template.
- **Screen templates:** The Screen template is responsible for the layout and the content of the screen, the portion of the portal page containing the output of the portlets.
- **Component templates:** Component templates are responsible for rendering the component itself and for starting the rendering of its children components. The children of container components (row and column) may be other containers or controls. The child of a control will always be a single portlet.

Page aggregation processing

The rendering process is a domino process starting with the root container. The root container triggers the rendering of all the child components in the page hierarchy as shown in Figure 10-9.

Rendering the screen triggers the aggregation of a page and its portlets. The *pageRender* tag in the screen starts the rendering process. If no portlet is maximized, then the *pageRender* tag calls the RootContainer.

The Root Container holds all the containers and controls for this page. The pageRender tag tells the Root Container to invoke the rendering of all its children. Since the Root Container is used only as a starting point for the aggregation, it does not render itself and therefore is not visible on the screen.

Each child of the Root Container invokes its template which is responsible for rendering all the content of its child. If the child contains further child components the *componentLoop* and *componentRender* tags execute the rendering of all these components one at a time.

Each component invokes its own template which in turn invokes more components until the leaf of each branch is reached. Thus, control moves between component classes and their respective JSPs. Each component writes its content to the servlet output stream.

When a control is reached, it invokes the rendering of the portlet, which adds its output to the output stream via its rendering. When the entire tree has been traversed, the output stream is closed and the output is sent back to the requesting client.

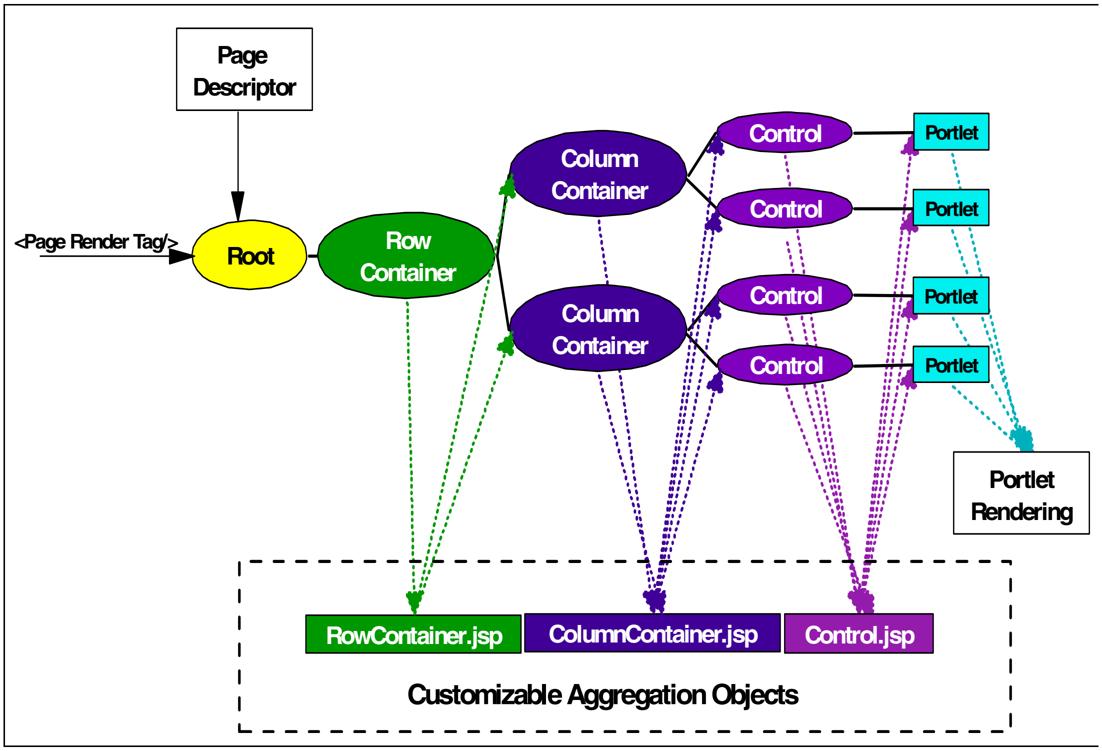


Figure 10-9 Page aggregation

10.4 Portlet solution patterns

Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems are excellent candidates for portlets because efficient, personalized access to these functions provide measurable returns on your portal investment. WebSphere Portal includes portlets that help you access a variety of ERP and CRM systems. At the time this book is written the portlet catalog on the IBM Portal Web site contains 55 portlets in the ERP section alone.

Enterprise Applications running on a backend or host system are another group of candidates for portlets, especially when the portal addresses the business-to-employee pattern and you want to provide a common working environment to your users, whatever application and system they may need for their work.

There are many ways to perform application integration in a Web environment. Not all of them are based on portlets and amongst the portlet-based solutions, several different architectural approaches can be applied. Depending on technical circumstances, the given time frame and the goals of the integration, typically different approaches may be combined in one portal solution.

We try to list some of the patterns you might think of. One way we can differentiate is along the line of shrink-wrapped versus roll-your-own.

Customizable portlets from a vendor

In this pattern, a portlet is provided that can be installed in Portlet Server and, after a configuration effort, the system or application in question can be accessed through the portal. Often, such a portlet is delivered by the vendor of the system that should be accessed. Both the Host On-Demand portlet and the Host Publisher portlet we use in the following examples are of this type.

Custom developed portlets

This pattern comes into play when either no vendor offers a portlet for the requested application, or the requested level of functionality, usability, accessibility or security is not met by the existing portlets. Another reason might be that you want to combine information or functionality of multiple applications seamlessly into one portlet.

Most probably, this integration will include using the Java Connector Architecture (JCA). JCA is a standard architecture for integrating J2EE applications with Enterprise Information Systems that are not relational databases. Each of these systems provides native APIs for identifying a function to call, specifying its input data, and processing its output data. The goal of the JCA is to achieve an independent API for coding these functions.

JCA also defines a standard Service Provider Interface (SPI) for integrating the transaction, security and connection management facilities of an application server with those of a transactional resource manager. Thus, JCA is a standards-based approach to managing connections, transactions, and secure access to enterprise application systems. IBM’s JCA connectors provide access to systems such as SAP, People Soft, CICS, and IMS. Leveraging its CrossWorlds® acquisition, IBM will also develop and integrate JCA connectors to many other systems.

Another way to look at portlets for application integration is from a topology point of view.

Client to remote application

In this pattern, used by IBM Host On-Demand, the portlet is just a bootstrap to allow the client to get in touch with the requested system or application, and Portal Server is the framework for the user interface. This implies that normally, an applet is involved which makes a direct network connection to a remote system. See Figure 10-10.

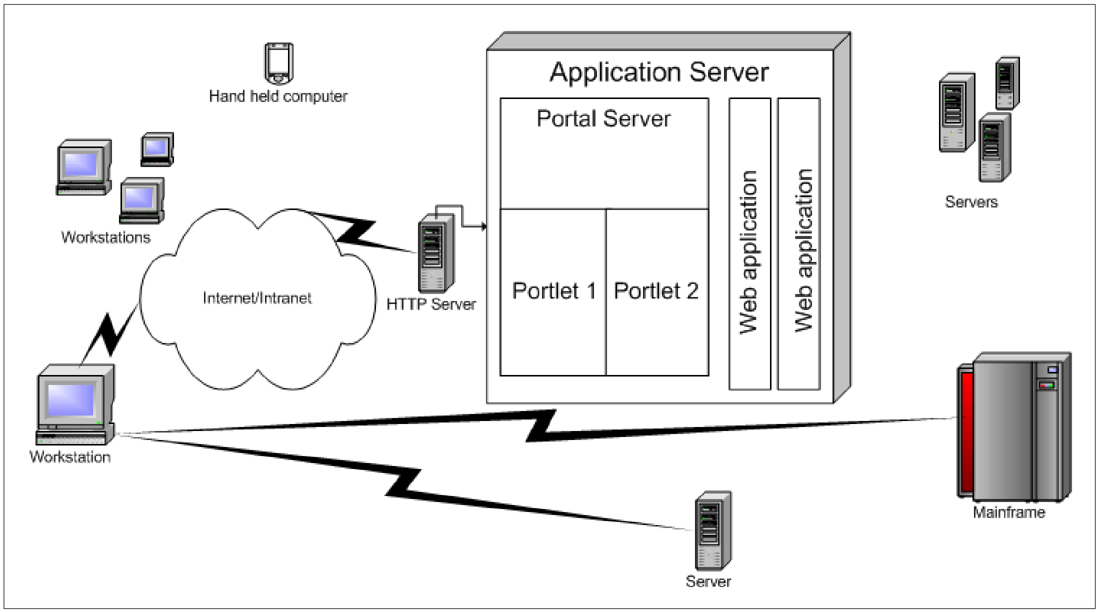


Figure 10-10 Portal Solutions - client to remote application

Portlet to remote application

This is the topology most likely used if you write your own application integration portlet. Access to the requested application or information is gained through standardized interfaces such as JCA connectors, JDBC and JMS, or by using a proprietary API provided by the application that is to be integrated (for example SAP Business Connector). See Figure 10-11.

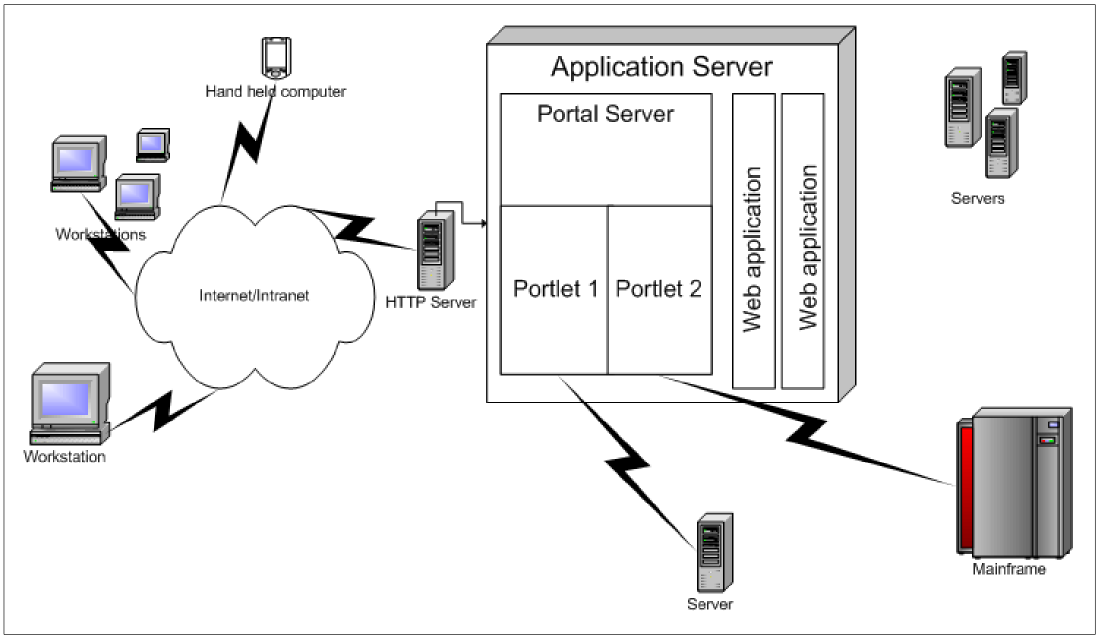


Figure 10-11 Portal Solutions - portlet to remote application

Portlet to Web application

In this pattern, most of the work is done in a Web application. Also, if you write a Web application using the JCA or EJB and create a portlet interface to it, you follow this pattern. See Figure 10-12.

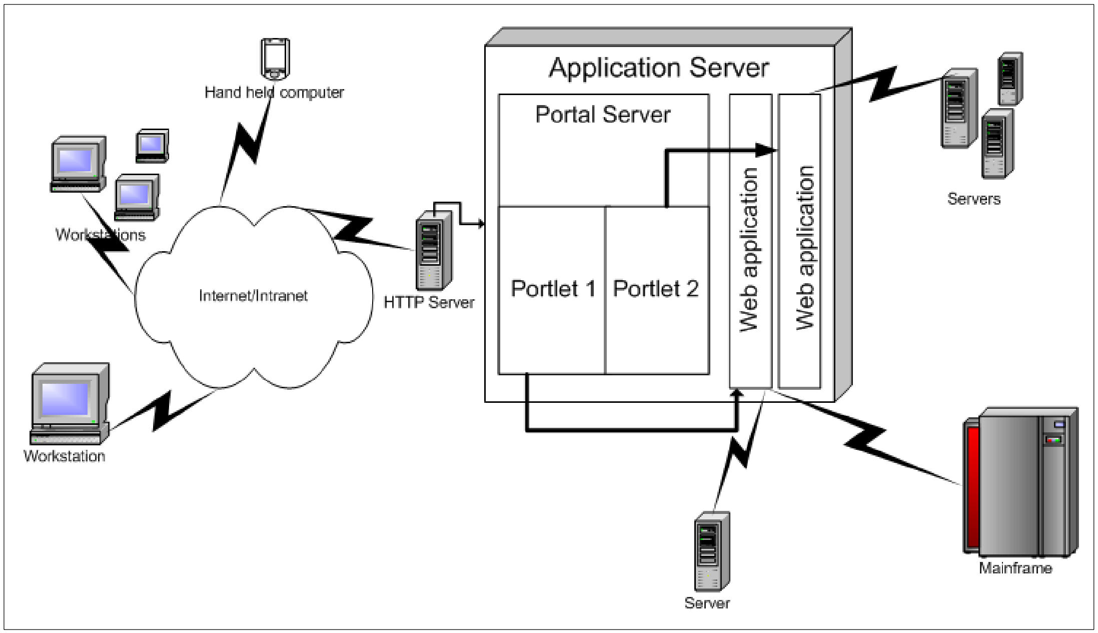


Figure 10-12 Portal Solutions - Portlet to Web Application

10.5 IDS and WebSphere portal server

WebSphere Portal uses databases to store various types of information. All database software must be installed, configured and running prior to the Portal Server installation if you:

- Have pre-existing databases
- Use third party databases

If you do not have existing databases, WebSphere Portal includes DB2 and Setup Manager can install it for you. If you perform the Standard DB2 installation with Setup Manager, several DB2 usernames and groups are automatically created on the system. It is recommended that you not alter these usernames and groups.

Portal Server maintains information about portal users in two databases: the Portal Server database (*wps*) and the Member Services database (*wms*).

You can use database software that is local or remote to the Portal Server.

The Portal Server and Member Services databases can be created and initialized during the Portal Server installation, or you can create these databases prior to the Portal Server installation and then initialize them during the Portal Server installation.

If you have uninstalled Portal Server, you cannot re-use the Portal Server and Member Services databases when you reinstall Portal Server. You must remove the original databases from the server and create the databases again before reinstalling Portal Server.

10.5.1 Configuring IDS for WebSphere Portal

Since version 4.2.1 of the WebSphere portal server you can use Informix IDS 9.3 or later as the database software for WebSphere Portal. You must install and configure IDS 9 prior to installing WebSphere Portal.

The following prerequisite steps must be completed prior to installing WebSphere Portal:

- Install Informix IDS 9.30 or later.
- Install Informix JDBC driver version 2.21.JC3 or later (2.21.JC4 if you want to use IDS 9.40!).
- Create a default Smart BLOB space.

Creating a default Smart BLOB space

If you install on a non-Windows platform, Informix does not create a default Smart BLOB space by default during installation. You must create a Smart BLOB space before installing WebSphere Portal. If you do not create a Smart BLOB space, no data exists in Portal Server after installation. Use the following steps to create a default Smart BLOB space for IDS 9.

1. Create an empty file (for example, /INFXDATA/sbspace001) using the following steps:
   - As user informix, change directories (for example, /INFXDATA) to where you will create the Smart BLOB space.
   - Execute `touch sbspace001`.
   - Execute `chmod 660 sbspace001`.
   - Make sure that the new file is owned by user informix and group informix.
2. Create the new Smart BLOB space by running the onspaces command:

```
onspaces -c -S <blobspace name> -p <path_to_blobspace> -o 0 -s <size>
```

In our example above:

```
onspaces -c -S sbspace001 -p /INFXDATA/sbspace001 -o 0 -s 102400
```

3. Make sure that the Smart BLOB space has been created correctly by using the onstat -d command. You should see the just added Smart BLOB space.
4. Now edit the `$ONCONFIG` file and modify the `SBSPACENAME` entry by adding the name of the new Smart BLOB space (for example, sbspace001).

Create an IDS 9 database with logging

Before we can create the portal server database, set an environment variable to enable UTF8 character-set support in the IDS database using the following command:

On Windows:

```
set DB_LOCALE=EN_US.UTF8
```

On UNIX:

```
DB_LOCALE=EN_US.UTF8
export DB_LOCALE
```

Now let us create the database itself. You can either use dbaccess in interactive mode, or write a small SQL script file. The script file should contain the following Informix SQL statement to create a new database with logging enabled:

```
CREATE DATABASE 'WPS' WITH LOG;
```

Tip: You can choose any database name you want. During installation of the WebSphere portal server you will be asked which database should be used. The suggested default name by the WPS Setup tool is *wps*.

Be sure that the environment variable DB_LOCALE is set correctly (see above), and execute the SQL script by running the following command:

```
dbaccess - <SQL script file>
```

where `<SQL script file>` is the name of the previously saved file.

Granting access rights for portal administrators

Portal administrators must be given DBA rights on the Informix database. Use these steps to grant access to portal administrators:

Start dbaccess and execute the following SQL statement:

```
grant dba to wpsadmin
```

In this statement, `wpsadmin` is the user name for the portal administrator.

Now you have an IDS 9 database prepared for portal server usage. You can now continue with the WebSphere portal server installation.

Tip: Before you continue with the WPS installation, take a note of some critical IDS 9 database connectivity parameters like, SERVERNAME, hostname, port number and database name. You will need this information during the portal server setup.

Summary

The WebSphere portal server family allows you to easily integrate your enterprise wide applications into a powerful frontend environment while supporting different clients at the same time. Informix IDS 9 is an optimal foundation for setting up such an environment trough its high performance, reliability and serviceability. You could even use IDS 9 high availability features like HDR (High Availability Data Replication) to increase the availability of your overall WebSphere portal / IDS solution without the need to buy additional components.

11

WebSphere MQ, messaging, and IDS

WebSphere MQ messaging products enable application integration by helping business applications to exchange information across different platforms, sending and receiving data as messages. Informix IDS is already supported on some WebSphere MQ platforms and will be another tier-1 database for future MQ releases. In addition, there is going to be a DataBlade API available for MQ access through the IBM alphaworks Web site.

In this chapter, the following topics are discussed:

- WebSphere MQ overview
- WebSphere and messaging
- MQSeries® transactional support for IDS
- IBM Informix MQSeries DataBlade

11.1 WebSphere MQ overview

IBM WebSphere® MQ (formerly MQSeries) is market-leading business integration software. It connects all your business software together to form one efficient enterprise by providing an open, scalable, industrial-strength messaging backbone.

WebSphere MQ minimizes time taken to integrate key resources and applications held in different systems, so your company can respond to the changing demands of e-business. By connecting business information with people and other applications, you can extract more value from existing investment, and quickly integrate new systems to support new market strategies.

WebSphere MQ features at a glance:

- Connect any commercial systems in business today (over 35 platforms supported)
- Ignore network disruptions – important data is always delivered
- Use less time and resources to become an e-business
- Exploit rich support from over 550 IBM Business Partners
- Allows business to integrate disparate islands of automation
- Time independent communication
- Assured one-time delivery
- WebSphere MQ supports high volume throughput, customer experience in excess of 250 million messages a day

Application programs and messaging

The IBM MQSeries range of products provides application programming services that enable application programs to communicate with each other using messages and queues. This form of communication is referred to as commercial messaging. It provides assured, once-only delivery of messages. Using MQSeries means that you can separate application programs, so that the program sending a message can continue processing without having to wait for a reply from the receiver. If the receiver, or the communication channel to it, is temporarily unavailable, the message can be forwarded at a later time. MQSeries also provides mechanisms for providing acknowledgements of messages received.

The programs that comprise an MQSeries application can be running on different computers, on different operating systems, and at different locations. The applications are written using a common programming interface known as the Message Queue Interface (MQI), so that applications developed on one platform can be transferred to another.

Figure 11-1 shows that when two applications communicate using messages and queues, one application puts a message on a queue, and the other application gets that message from the queue.

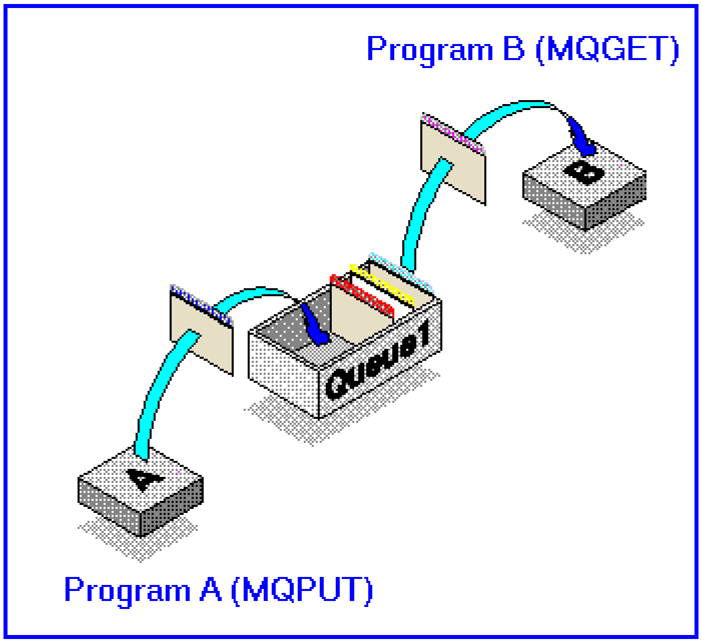


Figure 11-1 Two applications using a queue for communications

Queue managers

In MQSeries, queues are managed by a component called a Queue Manager. The queue manager provides messaging services for the applications and processes the MQI calls they issue. The queue manager ensures that messages are put on the correct queue or that they are routed to another queue manager.

Before applications can send any messages, you must create a queue manager and some queues. MQSeries for Windows provides the utilities to help you do this and to create any other MQSeries objects that you need for your applications.

How applications identify themselves to queue managers

Any MQSeries application must make a successful connection to a queue manager before it can make any other MQI calls. When the application successfully makes the connection, the queue manager returns a connection handle. This is an identifier that the application must specify each time it issues an MQI call. An application can connect to only one queue manager at a time (known as its local queue manager), so only one connection handle is valid (for that particular application) at a time. When the application has connected to a queue manager, all the MQI calls it issues are processed by that queue manager until it issues another MQI call to disconnect from that queue manager.

Opening a queue

Before your application can use a queue for messaging, it must open the queue. If you are putting a message on a queue, your application must open the queue for putting. Similarly, if you are getting a message from a queue, your application must open the queue for getting. You can specify that a queue is opened for both getting and putting, if required. The queue manager returns an object handle if the open request is successful. The application specifies this handle, together with the connection handle, when it issues a put or a get call. This ensures that the request is carried out on the correct queue.

Putting and getting messages

When the open request is confirmed, your application can put a message on the queue. To do this, it uses another MQI call on which you have to specify a number of parameters and data structures. These define all the information about the message you are putting, including the message type, its destination, which options are set, and so on. The message data (that is, the application-specific contents of the message your application is sending) is defined in a buffer, which you specify in the MQI call. When the queue manager processes the call, it adds a message descriptor, which contains information that is needed to ensure the message can be delivered properly. The message descriptor is in a format defined by MQSeries; the message data is defined by your application (this is what you put into the message data buffer in your application code).

The program that gets the messages from the queue must first open the queue for getting messages. It must then issue another MQI call to get the message from the queue. On this call, you have to specify which message you want to get.

Messaging using more than one queue manager

This arrangement is not typical for a real messaging application because both programs are running on the same computer, and connected to the same queue manager. In a commercial application, the putting and getting programs would probably be on different computers, and so connected to different queue managers.

Figure 11-2 shows how messaging works where the program putting the message and the program getting the message are on the different computers and are connected to different queue managers.

In this situation, you also need to create message channels to carry MQSeries messages between the queue managers.

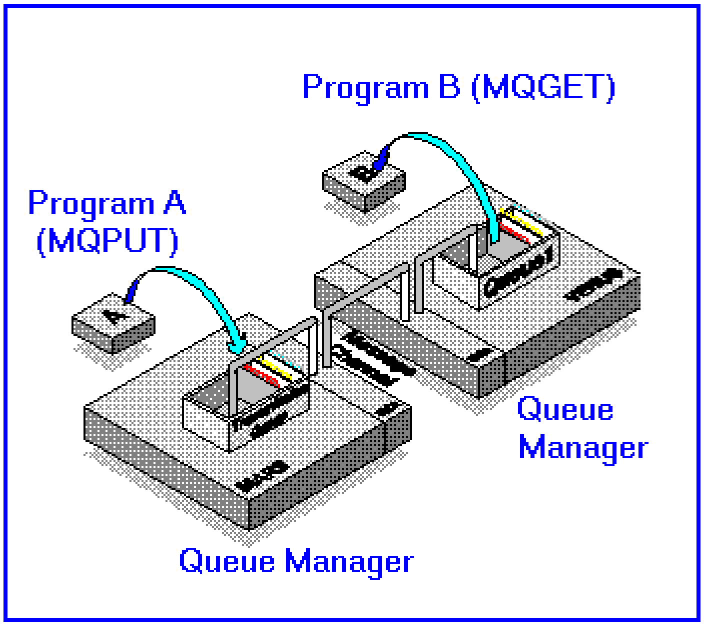


Figure 11-2 Applications connected to two different queue managers

11.2 WebSphere and messaging

In this section we discuss the role of WebSphere in messaging technologies.

11.2.1 Overview

Enterprises have been using messaging technologies for many years. Most uses of messaging with in application server environment can be categorized in the following scenarios.

- **Asynchronous communication:** Messaging provides a mechanism to for asynchronous communication between applications or application components. This could be point to point pattern or a publish/subscriber pattern explained in detail further down in the chapter.
- **Heterogeneous integration:** Enterprise systems, run with different hardware and different Operating Systems. They need a communication technology in common for integration and must use messaging for communication.
- **Temporary data storage:** Transaction monitor environments and other applications, require a temporary storage for the recovery in case of and even failure, but unlike an RDBMS the schema does not have to define a priori if messaging queues are used for the temporary data storage.

11.2.2 Java Message Services (JMS)

Provides a messaging API that allows for asynchronous communication between applications, though synchronous messaging is also possible

Asynchronous Messaging

JMS provides the ability for an application to send messages to another without waiting for a reply. There are two forms of messaging supported in JMS:

- **Point to Point Messaging:** Allows a peer-to-peer form of communication where two parties agree on a role of message producer and message consumer of a queue
- **Publish/Subscribe Messaging:** Allows for one-many communications pattern, where a message producer places messages into a topic based on the topic name. Subscribers who are interested in the topic can then subscribe to the topic. Subscribers are notified when a message for that topic arrives

11.2.3 WebSphere Messaging Engines

WebSphere provides support for a number of different messaging engines. They are:

- Lightweight JMS
- Embedded JMS
- External JMS/MQ

Lightweight JMS

The WebSphere Lightweight JMS engine provides messaging support for development tools WebSphere Studio Application Developer (WSAD) in particular. This is for rapid prototyping. WSAD uses the lightweight JMS Engine to provide messaging support with in the unit test environment. WSAD can also be used with any of the three engines mentioned before. The connection factories have to set up while configuring the test environment of WSAD for this.

The lightweight JMS engine is a close approximation of the J2EE required JMS functionality, but trades off support for security, persistence, and recoverability, for quick start up time. The lightweight JMS Engine does also support transactional semantics, but the transaction is not logged or recoverable after server restart.

Embedded JMS

The Embedded JMS Engine provides a subset of the WebSphere MQ 5.3.. It is a full J2EE compliant messaging engine. JMS provider management is integrated with WebSphere systems management. The message engine server is started and stopped along with the WebSphere Application Server. Topics and queues can be created in JMS resource configuration. The embedded JMS provider can only be installed if WebSphere MQ is not already installed on the node.

Some of the salient features of the Embedded JMS are listed here

- WebSphere MQ products provide "queue manager" and broker components, Embedded messaging provides a single server component (Embedded Messaging Server), accessed via a single integrated client component (Embedded Messaging client)
- Embedded Messaging is administered using the WebSphere Administration Console
- Can be accessed by thin Java clients via JNDI.
- Cannot interoperate with WebSphere MQ, WebSphere Event Broker, and not Java applications that do not have access to resources in the WebSphere application Server via JNDI
- When Embedded Messaging server is installed it is installed on a node A. All the WebSphere application Servers belonging to the cell to which the node is clustered must use the Embedded Messaging Server of node A, but thee participating messaging nodes need to have the Embedded Messaging Client installed.
- Embedded Messaging is integrated with the WebSphere application Security. UserIds are authenticated against the password in user registry, which the Application Server has been configured to use.
- The Embedded JMS Server starts and stops along with the WebSphere Application Server and has no visible separation of queue manager and broker components. It support point to point and Pub/Sub messaging patterns.

External JMS/MQ

The external JMS engine provides complete functionality for JMS messaging, including message queue clustering support. There can only be one WebSphere MQ JMS provider, embedded or external installed on a node.

Here are the possible modes of Installing WebSphere MQ:

- If WebSphere MQ is installed and then WebSphere Application Server is installed, then embedded JMS will not be installed, and the WebSphere MQ already installed will be used.
- If WebSphere Application Server is installed, then the embedded JMS will be automatically installed and can be upgraded to use the full WebSphere MQ if WebSphere MQ is installed subsequently.

The WebSphere MQ product family provides additional messaging support in the form of WebSphere MQ Integration Broker, which provides rule-based message transformation and routing that can be visually composed using message flows.

WAS, WebSphere MQ, and WebSphere MQ Event Broker

When WebSphere MQ and/or WebSphere MQ Event Broker with WebSphere application Server are integrated, the customer gets the benefit of these products with the J2EE applications. The effort is that messaging system has to be separately configured using the WebSphere MQ Event Broker tools like MQ Explorer, the connection between the Application Server and MQ or Event Broker. With in the Application Server tools, MQ and Event Broker are referred to collectively as the “WebSphere MQ JMS Provider”.

To use the “MQ JMS Provider” you install and configure either WebSphere MQ or WebSphere MQ Event broker and then configure the Application Server to connect to it instead of the Embedded Provider. The MQ products can be on the same or different server.

If your J2EE applications are using only point-to-point messaging, then WebSphere MQ is sufficient; if they are using publish/subscribe, then WebSphere MQ Event Broker is needed.

Here are some scenarios for which can use of the additional capabilities of WebSphere MQ, WebSphere MQ Event Broker, or WebSphere MQ Integrator Broker would be appropriate:

- A necessity to connect applications running in WebSphere Application Server with applications that use a wide selection of other language environments, runtime environments, and/or hardware platforms, or to connect to a large range of packaged applications that have either native a MQ interface, or for which an adapter is available.
- A necessity to support high message volumes (measured as a function of both message size and number of messages). With WebSphere MQ, Queue Manager Clustering can be used to distribute messaging workload across multiple queue managers.
- A requirement to decouple sending and receiving application environments, both from one another and from the underlying network that provides connectivity between them. Web-Sphere MQ message channels can allow the sending application to continue processing when the receiving application or its hardware is unavailable, and vice versa, and both applications may be able to continue operating when the network link is down.
- A requirement to support a large number of independent subscribers. With WebSphere MQ Event Broker, multiple brokers can be interconnected so as to form topology This allows publications to be “fanned out” and distributed across a large number of subscribing applications.
- A requirement to reuse existing WebSphere MQ or Event Broker infrastructure.

Other messaging providers

In general, WebSphere supports the use of MQ based JMS providers. Other JMS providers can be used from a client for non-managed connection to a JMS provider. These non-managed connections are obtained using the same programming model as WebSphere MQ based JMS connections, but they are not integrated to provide connection-pooling behavior.

11.3 WebSphere MQ Integration with IDS

There are currently two areas in which Informix IDS is supported in combination with WebSphere MQ: transactional support and a DataBlade API to the MQ queue manager. The following sections describe how to enable the different options in combination with IDS.

11.3.1 MQSeries Transactional Support for IDS

Transactional support for IDS and WebSphere MQ is currently available on the following platforms:

- MQSeries for HP-UX (10.20 or 11) and MQSeries for Sun Solaris - V5.2 plus CSD05 plus IY31724 or later running in conjunction with Informix IDS 7.3 or 9.21 or compatible version, or
- MQSeries for Sun Solaris, Intel Platform Edition - V5.1 running in conjunction with Informix IDS 9.21 or compatible version.

In addition to the MQ product itself, you also need to obtain the MQSeries SupportPac™ MC08. IBM employees can retrieve that SupportPac by using the following internal URL:

`http://www-3.ibm.com/software/integration/support/supportpacs/individual/mc08.html`

This SupportPac provides the sample programs and guidance necessary to allow MQSeries to co-ordinate Informix database updates and messaging activity in a single “unit of work”, providing transactionally assured updates to both resources. The co-ordination by MQSeries of Informix databases is achieved using the XA protocol with MQSeries acting as the XA Transaction Manager and Informix acting as an XA compliant Resource Manager. Applications which include both MQSeries and SQL activity may use the MQBEGIN verb to start a “unit of work”. Subsequent MQSeries and SQL activity can be “committed” or “backed out” atomically, using MQCMIT or MQBACK verbs.

This function is an extension to the existing database message resource co-ordination functionality available in MQSeries V5.0 or later for DB2, Oracle and Sybase. The versions of Informix supported are V7.3 or V9.2 or compatible version for HP-UX and Sun Solaris and V9.21 for Sun Solaris, Intel Platform Edition.

Further information can be found in the following MQSeries publications:

- MQSeries System Administration - Chapter 14 (Transactional Support)
- MQSeries Application Programming Guide - Chapter 13 (Committing and backing out units of work)

This functionality offers a significant step toward tighter integration of MQSeries with the Informix database which, prior to this function, could only be achieved by using an external transaction manager or sophisticated application program. This new functionality offers significant savings in the implementation of a transaction monitor when transactional integrity is only required for database updates.

11.3.2 IBM Informix MQSeries DataBlade

The Informix MQSeries DataBlade allows IBM Informix database applications to asynchronously communicate with other MQSeries applications. For instance, the new functions provide a simple way for an IBM Informix database application to publish database events to remote MQSeries applications, initiate a workflow through the optional MQSeries Workflow product, or communicate with an existing application package with the optional MQSeries Integrator product. This work is modeled on IBM WebSphere interfaces provided with DB2® Universal Database Version 8.x. More information about this can be found at in the DB2 Information Center.

The DataBlade provides access to MQSeries queues via function calls or tables using IBM/Informix's Virtual Table Interface (VTI). The VTI binds tables to MQSeries queues creating transparent access to MQSeries objects via tables. By binding the table to a queue a SQL developer can access the queue as if it were a table, the most comfortable interface from a database developers perspective.

Example 11-1 illustrates how to insert into a queue, followed by the corresponding retrieval utilizing MQ DataBlade function calls.

Example 11-1 Insert message into the queue

```
execute function
           MQSend('AMT.SAMPLE.SERVICE',
                  'AMT.SAMPLE.POLICY',
        'Hello Queue');
(expression)
           1
1 row(s) retrieved.
```

Example 11-2 illustrates how to remove a message from the same queue.

Example 11-2 Remove a message from the same queue

```
execute function MQReceive('AMT.SAMPLE.SERVICE',
                           'AMT.SAMPLE.POLICY');

(expression)  Hello Queue

1 row(s) retrieved.
```

In addition to a functional interface, the Virtual Table Interface (VTI) can be established to MQ queue also. This enables a table definition to be bound to a queue, so when an application inserts data into the table, it is put to the MQSeries queue. Conversely, a select from the bound table results in a fetch being executed against the queue.

Inserting onto the queue using VTI looks as shown in Example 11-3.

Example 11-3 Inserting onto a queue

```
insert into vtiMQ (msg) values ('Hello World!');

1 row(s) inserted.
```

Retrieval from a queue using the table interface looks as shown in Example 11-4.

Example 11-4 Retrieval from a queue

```
select msg from vtiMQ;

msg  Hello World!

1 row(s) retrieved.
```

For further information regarding the IBM Informix MQSeries DataBlade and how to obtain a copy of it, please refer to an upcoming section on the IBM alphaworks Web site: (http://www.alphaworks.ibm.com).

Conclusion

WebSphere MQ provides a consistent multiplatform, application-programming interface. A key factor is time-independent processing. This means that messages are dealt with promptly, even if one or more recipients are temporarily unavailable.

WebSphere MQ takes care of network interfaces, assures “once and once only” delivery of messages, deals with communications protocols, dynamically distributes workload across available resources, handles recovery after system problems, and helps make programs portable. Thus, programmers can use their skills to handle key business requirements, instead of wrestling with underlying network complexities.

The already existing support and upcoming enhancements for Informix IDS/WebSphere MQ opens new, powerful business integration options for Informix developers.

12

IBM Informix 4GL and WebSphere

The Informix 4GL has been one of the first Informix products in the market and has been very successful for more than a decade with Informix database developers worldwide. Many enterprise critical applications still rely on 4GL, and millions of lines of 4GL code are currently being used at thousands of customer sites.

IBM plans to address the needs of those 4GL developers/applications through different approaches. In addition to opening the Informix 4GL to the DB2 database family, there will be an evolutionary migration path into the world of J2EE environments through the integration into the WebSphere development environment (WSAD).

In this chapter we focus on the current plans for a 4GL integration into the world of WebSphere.

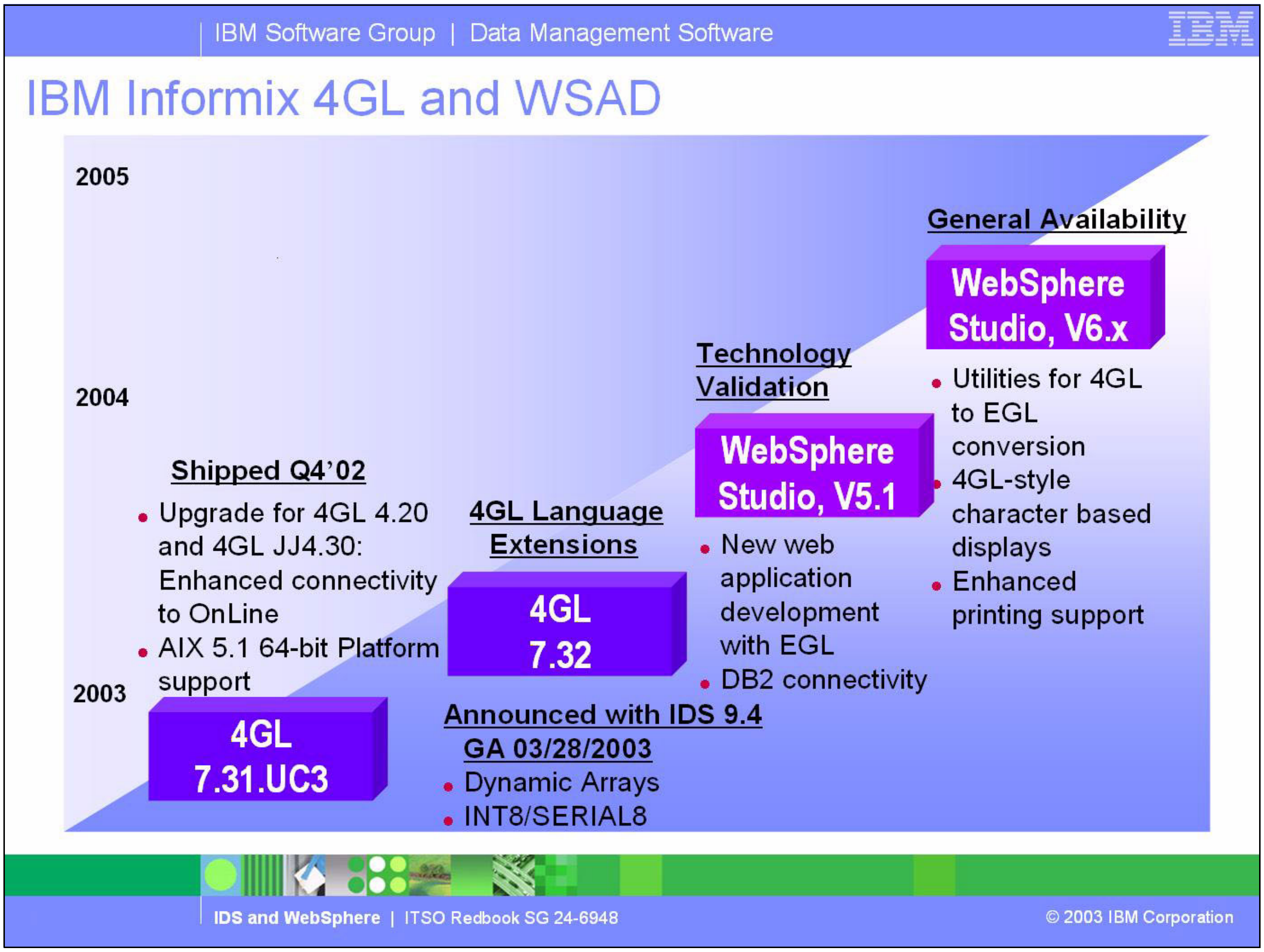
12.1 IBM Informix 4GL: Protecting your investment

In this section we consider the new capabilities of this product.

12.1.1 Informix 4GL

IBM Informix 4GL (I4GL) is an application development product that has been around for many years. As a very powerful and productivity improving development tool, it is heavily used across much of the Informix customer base and has become a key piece of development infrastructure in the Informix developer community. However, in today's Internet economy, the task of the application developer is rapidly changing. There are new functions and features required, particularly to support Internet-based applications.

For example, many companies want their employees to access even the mission-critical applications via a Web browser, or to run them on a PC on the client's desktop without abandoning the GUI interface available in the majority of programs. But, you have made a significant investment in I4GL and need to enhance that investment, not redevelop it. So what is the strategy and direction for I4GL? IBM has been busily putting together plans to protect the investment that you have in I4GL and to deliver the new capabilities that are required. So, how are we going to do that? Take a look at Figure 12-1.



IBM Software Group | Data Management Software

IBM Informix 4GL and WSAD

Shipped Q4'02

- Upgrade for 4GL 4.20 and 4GL JJ4.30: Enhanced connectivity to OnLine
- AIX 5.1 64-bit Platform support

4GL 7.31.UC3

4GL Language Extensions

4GL 7.32

Announced with IDS 9.4
GA 03/28/2003

- Dynamic Arrays
- INT8/SERIAL8

Technology Validation

WebSphere Studio, V5.1

- New web application development with EGL
- DB2 connectivity

General Availability

WebSphere Studio, V6.x

- Utilities for 4GL to EGL conversion
- 4GL-style character based displays
- Enhanced printing support

IDS and WebSphere | ITSO Redbook SG 24-6948

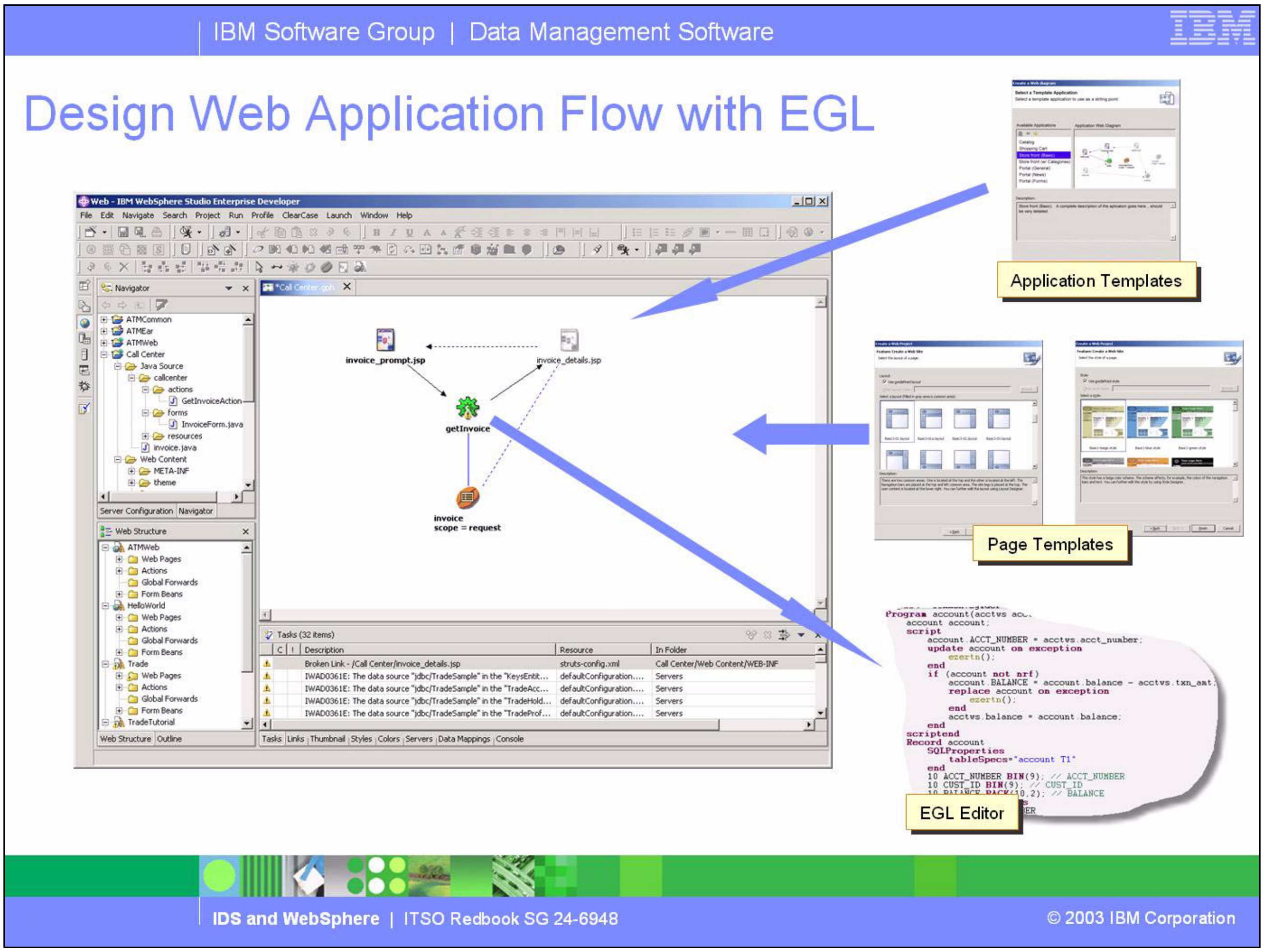
© 2003 IBM Corporation

Figure 12-1 The 4GL Roadmap for WebSphere integration

12.1.2 EGL and WebSphere

IBM WebSphere has a product called Enterprise Generation Language (EGL), which is a product that enables customers using the VisualAge Generator (VAGen) to move their applications into the WebSphere environment. One of the key benefits of this is that customers get a very powerful GUI oriented application development environment and one that puts them right into the mainstream of the IBM application development strategy. For example, it means they are now positioned to use, and will be integrated with, the WebSphere Studio Application Developer tools. That means they have access to screen painters and the entire integrated development environment, enabling the applications to be deployed as is typical with their traditional applications. But, it also provides the mechanisms to enable their applications to be deployed to the WebSphere Application Server environment. This means the can also be used in the Internet environment.

This is a very exciting product direction that should be of great interest to the I4GL user community. The direction then is to modify EGL so it will be easy to migrate from I4GL. Then the presence of tools, such as screen painters to handle forms, compilers generating Java code, and the ability to integrate with WebSphere at run-time, is very appealing. How are we going to do that? Consider Figure 12-2.



IBM Software Group | Data Management Software

Design Web Application Flow with EGL

Application Templates

Page Templates

EGL Editor

```
Program account(acctws acc
    account account;
    script
        account.ACCT_NUMBER = acctws.acct_number;
        update account on exception
            ezertn();
        end
        if (account not nrf)
            account.BALANCE = account.balance - acctws.txn_amt;
            replace account on exception
                ezertn();
            end
            acctws.balance = account.balance;
        end
    scriptend
    Record account
        SQLProperties
            tableSpecs="account T1"
        end
        10 ACCT_NUMBER BIN(9); // ACCT_NUMBER
        10 CUST_ID BIN(9); // CUST_ID
        10 BALANCE PACK(10,2); // BALANCE
```

Tasks (32 items)

C	!	Description	Resource	In Folder
		Broken Link - /Call Center/invoice_details.jsp	struts-config.xml	Call Center/Web Content/WEB-INF
		IWAD0361E: The data source "jdbc/TradeSample" in the "KeysEntit...	defaultConfiguration....	Servers
		IWAD0361E: The data source "jdbc/TradeSample" in the "TradeAcc...	defaultConfiguration....	Servers
		IWAD0361E: The data source "jdbc/TradeSample" in the "TradeHold...	defaultConfiguration....	Servers
		IWAD0361E: The data source "jdbc/TradeSample" in the "TradeProf...	defaultConfiguration....	Servers

IDS and WebSphere | ITSO Redbook SG 24-6948

© 2003 IBM Corporation

Figure 12-2 A possible Web application development scenario with EGL

12.1.3 Extending EGL to support I4GL

IBM is in the process of modifying EGL so that it is possible to automatically translate I4GL code into the extended EGL. The EGL runtime environment will provide a character-based screen deployment option (TUI – or terminal user interface – as opposed to GUI or graphical user interface) which will function virtually identically to the I4GL runtime. However, once it is translated, developers have the added advantage of being able to enhance the translated I4GL code to take advantage of the GUI interfaces and Web deployment options that are available with EGL. This is a significant benefit to I4GL customers and a major step in protecting their application development investment.

This task has required a number of changes to EGL. For example, the database support in EGL had to be extended to accommodate I4GL. EGL was database agnostic; for example, it supports, data sources such as IMS and simple files as well as SQL databases, but it hid the SQL syntax from the developer. One of I4GL's strengths has always been the close integration of SQL with the programming language. However, it should be noted that the only dialect of SQL supported was the Informix version, and the only connectivity method was the Informix-proprietary ESQL/C. One side effect of better support for SQL in EGL, and for Informix's dialect of SQL in particular, was that the type system needed to be made richer. EGL already supports the 3GL constructs that I4GL supports, such as functions, for loops, while loops, if statements, and so on.

Type system

To achieve the goal of translating I4GL to EGL automatically, the EGL type system had to be revised to support essentially all the SQL data types – both those in standard SQL-92 and those in the Informix dialect of SQL. EGL already had support for character and numeric data types, but needed to include support for the date, time and interval types, as well as being refined to handle specialized types such as BLOB, CLOB, BYTE, TEXT, LVARCHAR, INT8, SERIAL, and SERIAL8.

Procedural statements

The EGL procedural statements are similar to those in I4GL, but there are some differences. For example, the END keyword is not followed by a keyword such as IF or WHILE, and the keyword LET is not necessary. However, omission of the LET keyword means that statements must be terminated by semi-colons. EGL functions have prototypes, which are unlike I4GL functions. This has implications for those I4GL functions that pass values of one type and expect the called function to convert the value on entry, or for functions which return a type that is normally converted into another when the return value is collected off the I4GL stack. EGL supports pass by reference semantics by default; whereas I4GL supports pass by value semantics. Reconciling these differences requires new keywords in EGL and care in converting the I4GL code. But, it is a function that can be automated.

SQL statements

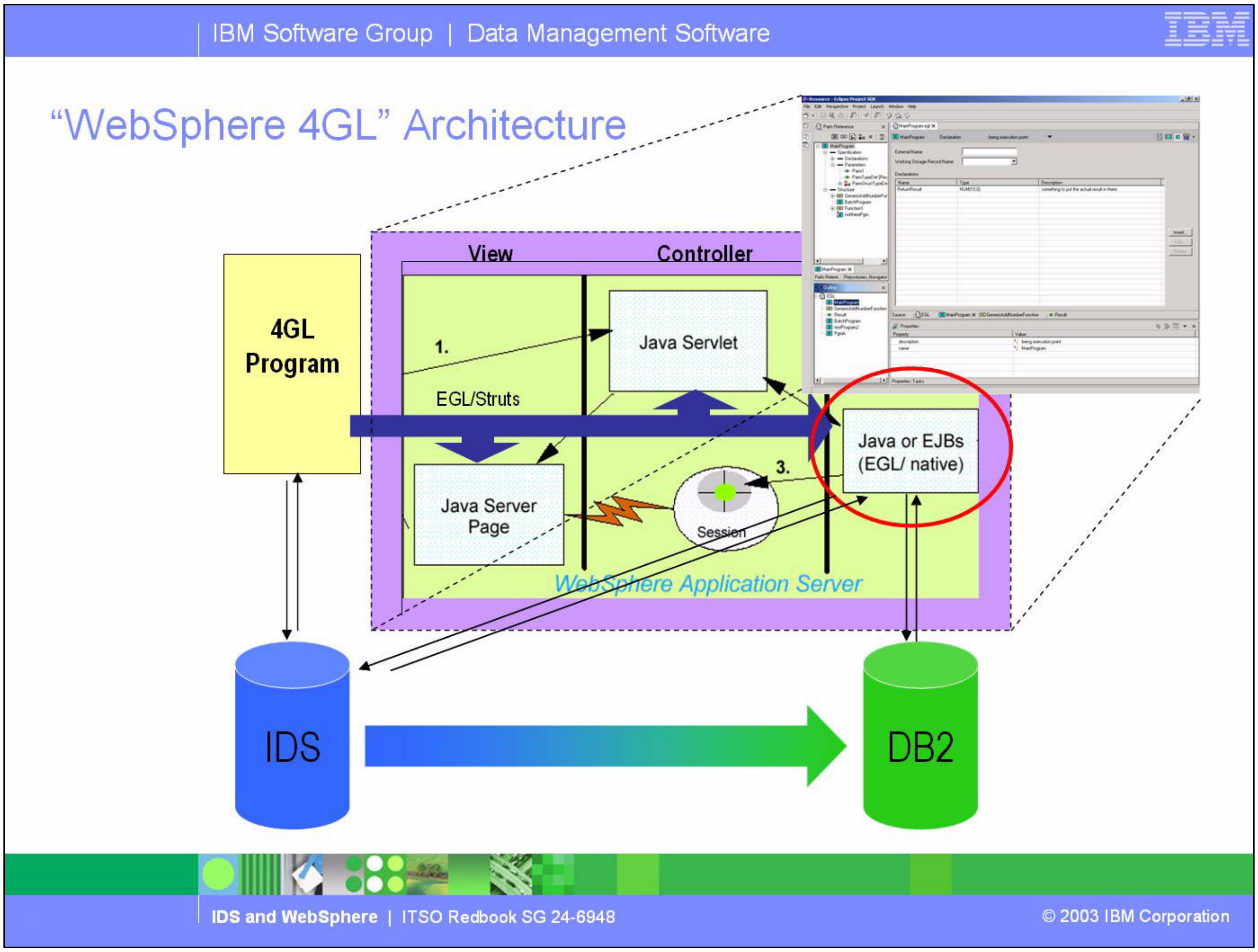
EGL will allow SQL statements to be written more nearly as in I4GL.

Non-procedural statements

I4GL provides a number of non-procedural statements, the key ones being MENU, INPUT, INPUT ARRAY, DISPLAY ARRAY, CONSTRUCT, and PROMPT. It is convenient to count DISPLAY in this group even though it doesn't involve user-interaction as the others do. These statements will be translated almost directly into EGL; that is, new EGL statements will be provided to support these I4GL operations. The keyword DISPLAY will become SHOW. All the others will continue to use their current names, and will retain their characteristic structure with identified events (such as BEFORE FIELD or AFTER INPUT or ON KEY) and the associated block of I4GL code being migrated to EGL. The code to be generated from these will be complex, but that's a problem for the EGL compilers and runtime, not the I4GL programmer.

Form files

I4GL form files will be translated into a new format that can be managed by an enhanced version of the EGL screen form painter. All the existing functionality will be retained, and valuable new features will be made available to applications, such as control over both the foreground and background colors on the screen. Further, the screen painter will allow for both TUI and GUI displays, so it will be a key help in migrating to the browser and GUI future. See Figure 12-3.



IBM Software Group | Data Management Software

"WebSphere 4GL" Architecture

IDS and WebSphere | ITSO Redbook SG 24-6948

© 2003 IBM Corporation

Figure 12-3 4GL/EGL interaction with the WebSphere environment

Reports

The I4GL report writing facilities are one of its most valuable features, even if they also show the age of the product. This is one of the areas where the exact solution to the problem has not be decided, but there are two primary alternatives. The preferred one is based on WebSphere as a whole incorporating Crystal Reports (from CrystalDecisions – http://www.crystaldecisions.com). This will allow EGL to use Crystal Reports as the report generator, and there would be a mechanism to extract the I4GL report formatting functions (REPORT…END REPORT) and convert this into a Crystal Report format, while the report driver code would accumulate the data and send it to the Crystal Report code.

The alternative is to incorporate much of the I4GL report logic directly into EGL. The bulk of the code would be translated verbatim; structures such as ON EVERY ROW blocks would continue to exist and so on. There would be analogous changes to the contents of the formatting blocks. The big advantage of Crystal Reports would be the vast range of output formats supported, including variable width fonts and HTML. No final decision has been made, pending commercial discussions.

12.1.4 Moving from I4GL to EGL

The expectation of this development activity is that the task of moving I4GL code to EGL, will be accomplished nearly automatically. There may be a few applications which use extremely obscure statements – maybe VALIDATE LIKE and SCROLL – that will not be migrated 100% automatically, but, that should represent a very small percentage of the total number of applications (and only a small part of those applications). There will be support for C functions, but the I4GL internals will be completely revised. The effect will be that anything in the C code that relies on the internals of the I4GL library will not work. Functions using the documented C interface functions should be reusable, but will need to be re-evaluated for continued relevance in the EGL environment.

One of the less easily automated aspects of the migration will be adapting the I4GL build process to EGL. The EGL build will require explicit knowledge of all functions (and probably form files) before compilation. The I4GL program design database does not record all the form files or message files associated with a program, so libraries will need careful evaluation. More typically, programmers use the MAKE program. Unfortunately, the makefiles used to compile I4GL are not standardized at all, so it will be hard to automatically convert them to handle EGL.

12.1.5 The value of WebSphere Application Developer

Once the I4GL applications have been translated into EGL, you will be able to take advantage of the WebSphere Application Developer. This gives you significantly more capability, with a state of the art application development environment. It will impact your application development productivity and thus the cost of application development and maintenance. And you are positioned to more easily extend your reach into the Internet environment. This is an essential strategy and direction to remain competitive in today's online and On-Demand environment.

13

Implementation hints and tips

We have taken you through the installation and configuration processes for IDS and WebSphere, which should make the implementation much easier for you. But, while installing and configuring the products, there is always the possibility to encounter problems. We encountered a few during our implementation and wanted to document them in the event you have the same experience. This will ensure that your implementation experience will be faster and easier, and that the products will give you their best performance.

13.1 Our implementation experience

We have included in this chapter descriptions of several problems encountered during our implementation process. Along with the description, we offer either a solution to the problem or an alternative approach that we selected. Our objective is to help you avoid these problems, or at least have a solution available if you do encounter them. We hope these hints and tips will help speed you along with your implementation experience.

13.1.1 Installing SuSE 8.0 Linux

We installed SuSE 8.0 Linux on an Intel server with 760 MB of memory. The installation of SuSE is simple and easy, and can be performed via FTP. We chose this type of installation and we followed the instructions found at the following site:

ftp://ftp.suse.com/pub/suse/i386/current/README.FTP

To install SuSE Linux from this site, follow these steps:

1. Go to ftp://ftp.suse.com/pub/suse/i386/<version>/disks, and download the floppy disk image called *bootdisk* as well as the module disk image files you need (at least module1 and the network modules from modules3). Write the images to floppy disks using the 'dd' command:

   ```
   dd if=[path_to_image] of=/dev/fd0 bs=36b
   ```

   On non-linux systems, use the rawrite utility from the dosutils/rawrite directory (rawrite.exe).

   We copied all three files (bootdisk, modules1 and modules3) for our implementation.

Attention: We first tried to install SuSE 8.1 using the created floppy disks, but after loading the second disk (modules1) the screen turned black and we could not continue. Then we tried the 8.0 and it worked fine.

2. Boot from CD/floppy and at the boot prompt enter the installation source:

   ```
   linux install=ftp://ftp_server/directory
   ```

   Remember to substitute “ftp_server” and “directory” with the appropriate values (such as install=ftp://ftp.suse.com/pub/suse/i386/8.0 if you are installing from the SuSE ftp server).

   Alternatively, choose “manual installation” and configure the network in the installation program.

Tip: You might want to use a SuSE mirror site. In our implementation we did, and it was much faster to download the installation files.

3. When the installation files are downloaded, you can begin the installation and configuration of SuSE. Follow the screen instructions to complete the installation.

Important: During the disk configuration, do not configure all the space as REISER file systems. For IDS it is important to use another type of filesystem, such as ext3.

VNC: Remote administration

In our installation we used VNC (Virtual Network Computing) to emulate the X-window environment of SuSE. This proved to be very effective and easy to use. You can download both the VNC server and client from:

http://www.uk.research.att.com/vnc/

You must install the VNC server on the Linux server and then install the VNC viewer on your remote Windows processor.

VNC server

1. Copy the compressed file on a temporary directory, for example **/tmp**
2. Uncompress the file and extract the installation files. We found the file downloaded were in a tar format, so we ran:

   ```
   tar -xvf vnc-3.3.7-x86_linux.tar
   ```

3. This command creates a new directory called **vnc-3.3.7-x86_linux**. Just enter the directory and run:

   ```
   ./vncinstall <installation directory>
   ```

   We chose the **/opt/vnc** directory

4. Go to the installation directory and run the following command to enable the VNC server:

   ```
   cd /opt/vnc
   ./vncserver
   ```

   This will start the server and the VNC viewer can emulate the X-window environment of SuSE.

   The initialization process asks for a password that needs to be filled out when the viewer is trying to access the server.

   It shows the display number that needs to be used by the viewer and also shows the paths of its message log files.

> **Attention:** Once the VNC server is started, its message log is displayed on the screen. Open the log and you will notice that the vnc server is listening to a specific TCP/IP port. This port also needs to be configured in the firewall if you choose to leave it enabled.

VNC viewer

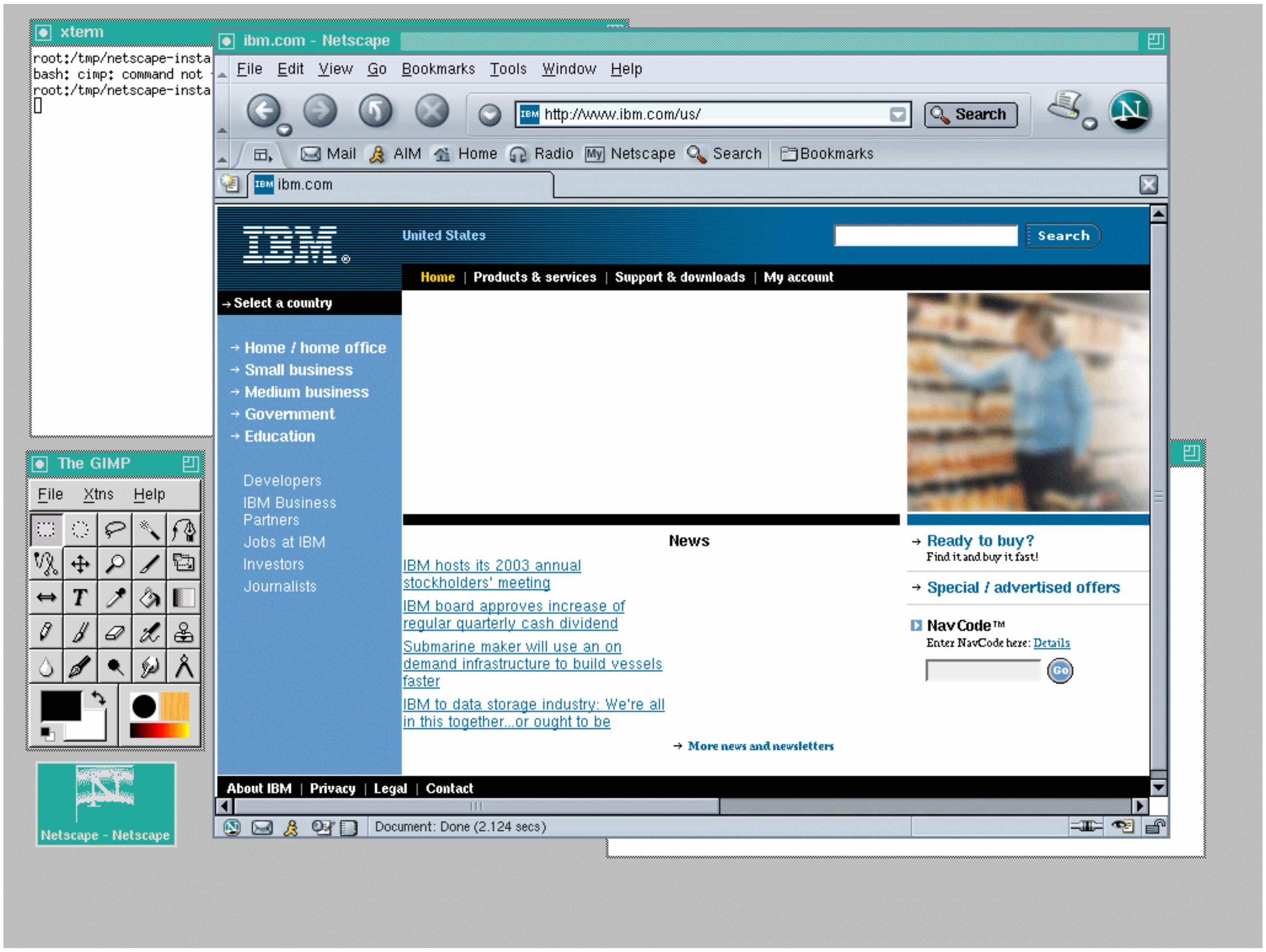
The viewer installation is even easier, just uncompress the file using Winzip on your Windows processor and follow the instructions.

Once the viewer is installed you simply need to execute the program by double-clicking its icon and provide the necessary information, such as:

Host name and display: The Linux host name and the display number showed when the VNC server was started.

Password: The password that was provided when the vnc server was configured.

If everything is correct, you should see a window as shown in Figure 13-1.



xterm
ibm.com - Netscape
File Edit View Go Bookmarks Tools Window Help
http://www.ibm.com/us/
Search
Mail AIM Home Radio Netscape Search Bookmarks
ibm.com
IBM
United States
Search
Home | Products & services | Support & downloads | My account
Select a country
Home / home office
Small business
Medium business
Government
Education
Developers
IBM Business Partners
Jobs at IBM
Investors
Journalists
News
IBM hosts its 2003 annual stockholders' meeting
IBM board approves increase of regular quarterly cash dividend
Submarine maker will use an on demand infrastructure to build vessels faster
IBM to data storage industry: We're all in this together...or ought to be
More news and newsletters
Ready to buy?
Find it and buy it fast!
Special / advertised offers
NavCode™
Enter NavCode here: Details
Go
About IBM | Privacy | Legal | Contact
Document: Done (2.124 secs)
The GIMP
File Xtns Help
Netscape - Netscape

```
root:/tmp/netscape-insta
bash: cimp: command not
root:/tmp/netscape-insta
```

Figure 13-1 X-window emulation through VNC

13.1.2 Installing IDS on SuSE 8.0 Linux

In this section we describe some of the issues we had while installing IDS on SuSE 8.0, and explain the solutions we used for them.

IDS installation

When using IDS 9.40 with LINUX systems there are two points to consider:

1. When using SUSE Linux 8.1 (and United Linux), we needed a glibc patch before IDS 9.40 would initialize. For SuSe Linux 8.1 this patch can be obtained from

   ftp://ftp.suse.com/pub/suse/i386/update/8.1/rpm/i686/glibc-2.2.5-165.i686.rpm

You need to install the patch with the -U option in the rpm command:

```
rpm -U glibc-2.2.5-165.i686.rpm
```

Attention: When trying to install the new glibc, a dependency error is reported indicating that we need to install a *filesystem* package before installing the glibc. You can find this package at:

```
ftp://ftp.suse.com/pub/suse/i386/8.1/suse/i586/filesystem-2002.9.2-50.i586.rpm
```

Install this package (rpm -iv filesystem-2002.9.2-50.i586.rpm), then the new glibc will be installed with no problems.

Notice that both packages were taken from the 8.1 distribution, but they worked fine on SuSE 8.0 as well.

This is due to the following information found in the machine notes ($INFORMIXDIR/release/en_us/0333/ids_machine_notes_9.40.txt)

This product was built on RedHat Linux 7.2 for i686 compatible processors and is targeted for Linux Kernel 2.4.7 or higher and glibc 2.2.4 or higher versions.

So, this means that IDS 9.40 was built on a determined Linux kernel and glibc version. The product is expected to work on newer Linux versions, but sometimes problems can occur.

Without the patch, the initialization will fail. When doing an oninit -ivy, the last message is:

```
Bringing up ADM VP...
```

The initialization hangs, and an **`onstat -`** shows that the server is not up.

Important: Take care not to put the chunks onto a REISER file system. This will make the instance very slow! So the solution is to put the chunks onto a different file system, such as ext3.

Firewall on SuSE

During the installation process of SuSE, you have the option to enable a firewall. If you choose to utilize the SuSE firewall, you may later have problems using both IDS and WebSphere. In our installation, not even the ftp and telnet ports were open.

You may also encounter problems when trying to connect to IDS from a client processor using TCP/IP. You will have to manually add the port number (in our case 1533) configured for IDS on the firewall. As for WebSphere Application Server, you may find problems using the administration console, because the 9090 port is not configured in the firewall.

To solve these issues, you can either configure these extra TCP/IP ports in the firewall or you can simply disable the firewall during your tests. For both options, you can use the YaST control center to configure the firewall (System ->YaST2).

Here we show how to manually add some new ports in the firewall configuration:

1. First open the YaST control center using the programs menu or using the shortcut to the control center in the toolbar. If you are not logged as *root* a window asking for the *root* password is displayed. See Figure 13-2.

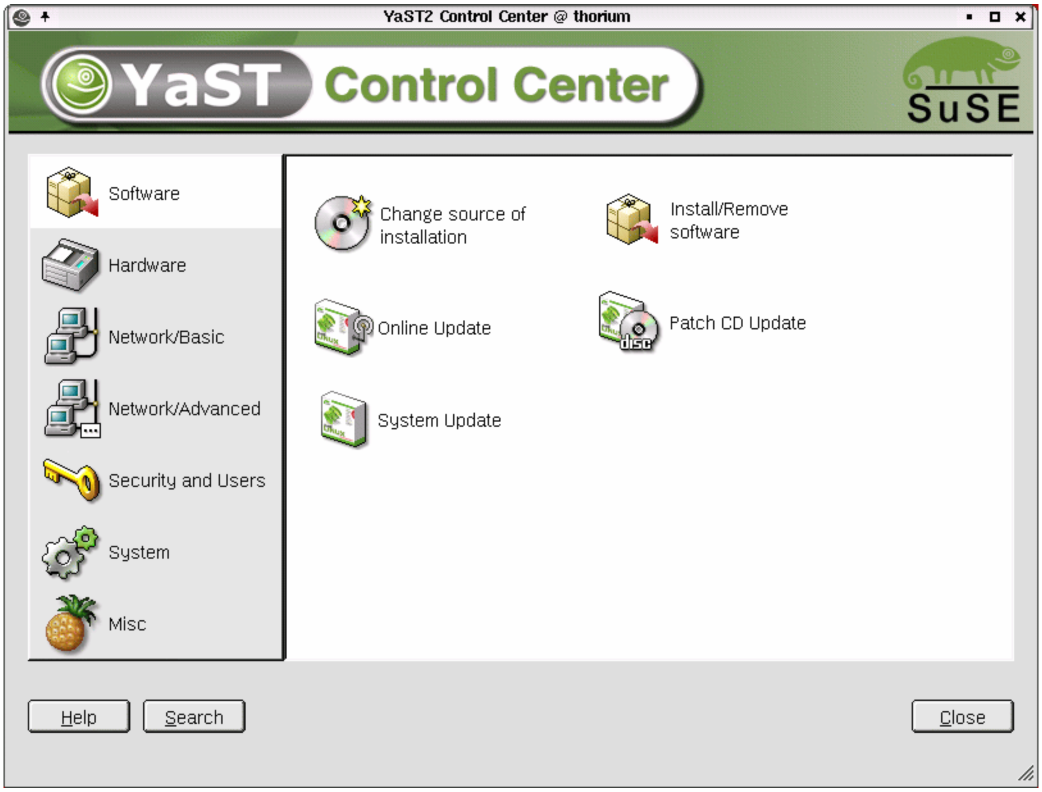
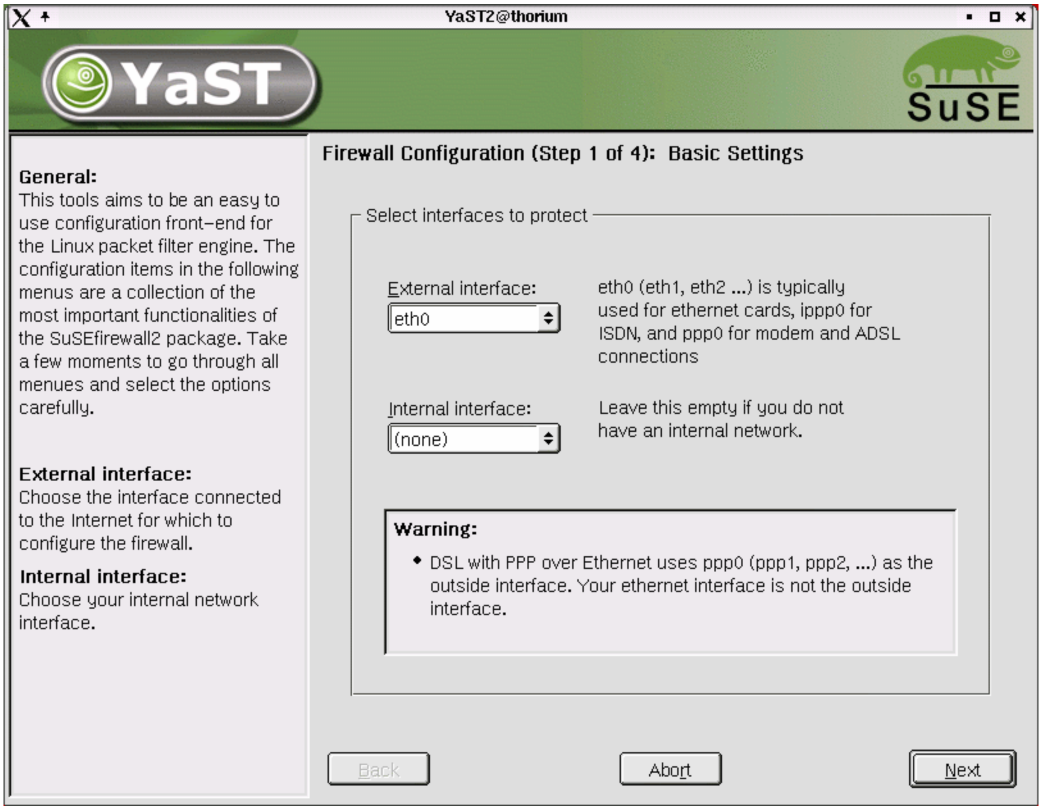


Figure 13-2 YaST Control Center

2. Choose the **Security and Users** option and then choose **Firewall**.

3. The control center uses four steps to configure the network. First you need to choose the device that you want to configure. Usually for network cards the device is **eth0**. Click **Next**. See Figure 13-3.



General:
This tools aims to be an easy to use configuration front–end for the Linux packet filter engine. The configuration items in the following menus are a collection of the most important functionalities of the SuSEfirewall2 package. Take a few moments to go through all menues and select the options carefully.

External interface:
Choose the interface connected to the Internet for which to configure the firewall.

Internal interface:
Choose your internal network interface.

Firewall Configuration (Step 1 of 4): Basic Settings

Select interfaces to protect

External interface: eth0

eth0 (eth1, eth2 ...) is typically used for ethernet cards, ippp0 for ISDN, and ppp0 for modem and ADSL connections

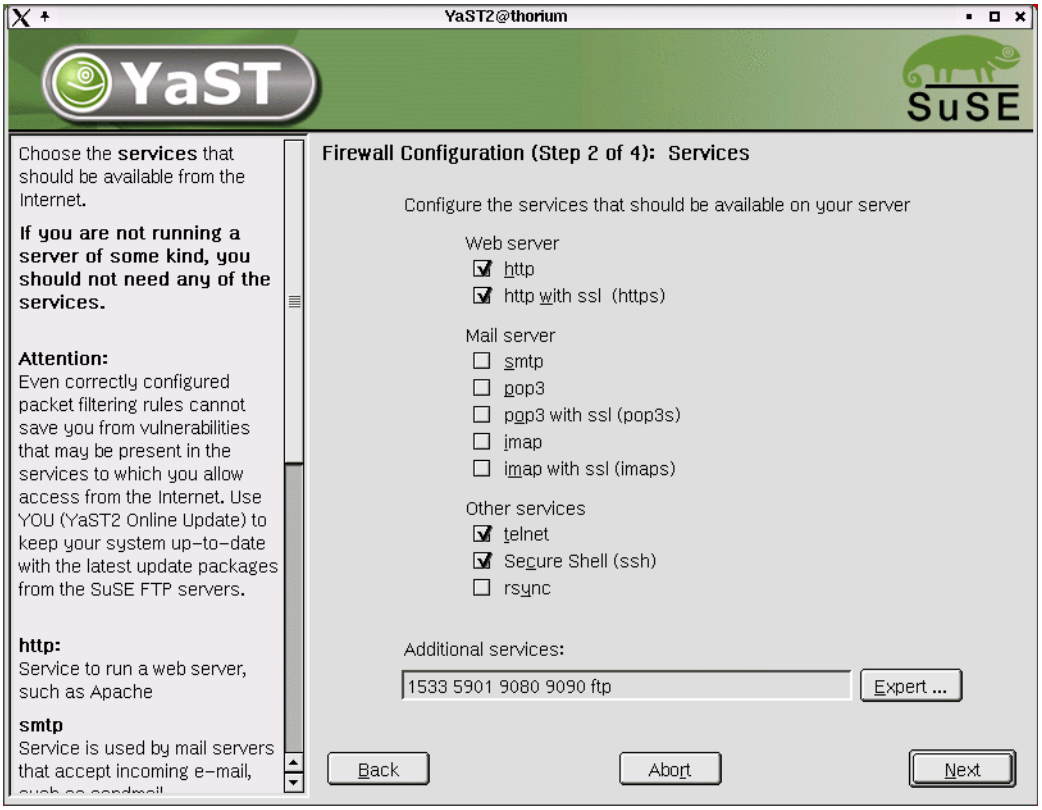
Internal interface: (none)

Leave this empty if you do not have an internal network.

Warning:
- DSL with PPP over Ethernet uses ppp0 (ppp1, ppp2, ...) as the outside interface. Your ethernet interface is not the outside interface.

Figure 13-3 List of network devices

4. On the next screen the configured ports are showed. Notice that there is a text field for additional services. To add a new service, click the **Expert...** button. See Figure 13-4.



Choose the **services** that should be available from the Internet.

If you are not running a server of some kind, you should not need any of the services.

Attention:
Even correctly configured packet filtering rules cannot save you from vulnerabilities that may be present in the services to which you allow access from the Internet. Use YOU (YaST2 Online Update) to keep your system up-to-date with the latest update packages from the SuSE FTP servers.

http:
Service to run a web server, such as Apache

smtp
Service is used by mail servers that accept incoming e-mail,

Firewall Configuration (Step 2 of 4): Services

Configure the services that should be available on your server

Web server
☑ http
☑ http with ssl (https)

Mail server
□ smtp
□ pop3
□ pop3 with ssl (pop3s)
□ imap
□ imap with ssl (imaps)

Other services
☑ telnet
☑ Secure Shell (ssh)
□ rsync

Additional services:
1533 5901 9080 9090 ftp

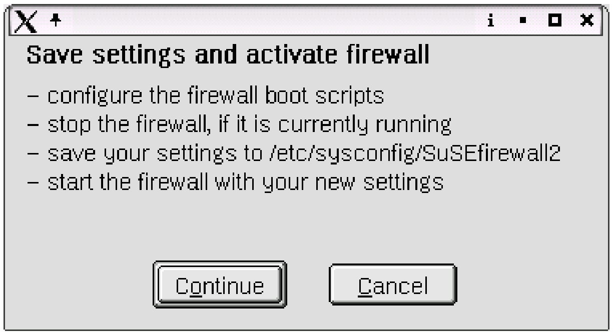
Figure 13-4 Extra services added

5. Add the TCP/IP port number that you need to be opened. For example, the IDS and WAS ports (1533, 9090, and 9080). Click next and the new added services should be displayed in the list. Click **Next**.

6. The next step shows some Features. Click **Next**.

7. The next screen shows some Logging options. Click **Next**.

8. Confirm your changes and activate the firewall. See Figure 13-5.

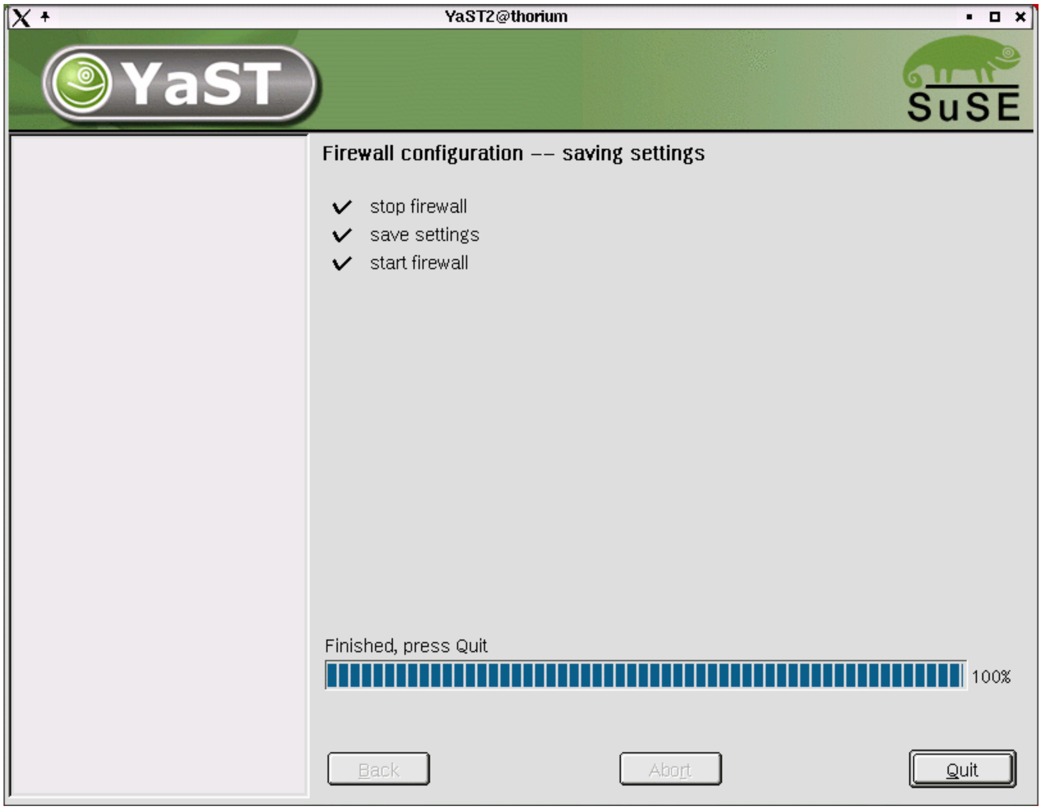


Save settings and activate firewall

– configure the firewall boot scripts
– stop the firewall, if it is currently running
– save your settings to /etc/sysconfig/SuSEfirewall2
– start the firewall with your new settings

Figure 13-5 Configuration confirmation

9. The firewall is now reconfigured. Just click the **Quit** button to exit the YaST control center. See Figure 13-6.



Firewall configuration -- saving settings

✓ stop firewall
✓ save settings
✓ start firewall

Finished, press Quit

100%

Figure 13-6 Firewall is reconfigured

13.1.3 Performance tuning guidelines

For purposes of this redbook, we only used IDS and WebSphere for testing purposes so there was no requirement for performance tuning. However, there are some general guidelines in this environment to be considered, and are presented here. This section is not intended to represent a complete or all-inclusive set of tuning guidelines.

From the IDS standpoint, the performance tuning process is very dependent on the systems environment and usually requires either a experienced server administrator or an IBM consultant to configure and test the installation. This is to be expected, particularly for the first implementation, since new applications and users may be added to the system. The performance tuning process for IDS should be re-analyzed and re-configured on a regular basis for best performance.

It is not within the scope of this book to describe how to tune an IDS server, but we present some general guidelines that could be used as a starting point. Normally these guidelines should work fine for new systems, but after some time more advanced techniques may be required. The guidelines are presented for two system types: OLTP and DSS.

OLTP tuning

When tuning the OnLine system for an OLTP environment it is important to focus on specific areas that will have the greatest impact on performance. What you want to achieve is:

- High read and write buffer cache rates
- Fast checkpoints
- Maximum I/O throughput by:
  - Eliminating I/O bottlenecks
  - Optimizing fragmentation strategy (including fragmenting the indexes as well as data)
  - Optimizing index utilization

Typically, OLTP environments are characterized by a large number of users performing a high volume of short transactions (INSERT, UPDATE and DELETE). When tuning IDS in an OLTP environment it is important to spend your time in areas that will have the greatest impact on performance. It is also important to realize that it is iterative in nature and doing too much change at one time can present a challenge. And finally, benchmarking is key to knowing where you stand and when you have success. With this in mind, here are some initial ONCONFIG ($INFORMIXDIR/etc/$ONCONFIG) settings for OLTP.

- BUFFERS - Maximize. Set to 50% - 75% of available free memory. Set to an even greater percent if not using the memory in the in virtual portion (after testing to see if the memory is needed as shown below). When tuning this parameter and SHMVIRTSIZE, understand that both may need to be changed if the combined total is 75% of the OS physical memory.
- LOCKS - Set to 1000 * number of users.
- PHYSBUFF - Pages per I/O should be about 75% of the buffer size (to monitor use onstat -l).
- LOGBUFF - Pages per I/O should be about 75% of the buffer size (to monitor use onstat -l).
- LRUS - Configure four LRU pairs per CPU VP.
- CLEANERS - Configure one page cleaner thread per LRU pair.
- SHMVIRTSIZE - Set to 32000 + expected number of users * 800
- CKPTINTVL - Set to 9999. Let the physical log initiate checkpoints. This may be contrary to popular belief but the reasoning is that you make the LRU cleaning keep dirty buffers to a minimum and even with a long interval, you can minimize the checkpoint waits.
- LRU_MAX_DIRTY - Set to 10. With many systems, the final settings may be as low as 1. Also with 9.4 release, you can use fractional percentages so the number of dirty buffers at checkpoint time should only be the maximum duration that system should perform a checkpoint in.
- LRU_MIN_DIRTY - Set to 5. With many systems, the final setting may be as low as 0. Also with 9.4 release, you can use fractional percentages so you should not need a 0.
- RA_PAGES - Set to 32.
- RA_THRESHOLD - Set to 30.

DSS tuning

When tuning the OnLine system for a Decision Support Systems (DSS) environment it is important to focus on specific areas that will have the greatest impact on performance. What you want to achieve are:

- Optimum memory utilization
- Parallel data queries (PDQ)
- Light scans
- Maximum I/O throughput

Optimum memory utilization: The area that will have the greatest impact is shared memory. Due to the nature of DSS queries large amounts of shared memory located in the virtual segment is required to perform a variety of operations, such as light scans, hash joins, and sorts. It is critical to properly configure and tune the shared memory and PDQ parameters in the OnLine onconfig file. To increase performance for DSS queries, increase the amount of available virtual shared memory. With this in mind here are initial ONCONFIG settings for tuning DSS queries:

- BUFFERS - Minimize. Set to 2000
- SHMVIRTSIZE - Maximize. Set to 75% of available memory (or an even higher % if memory not needed elsewhere).
- SHMADD - Set to 32000
- SHMTOTAL - Set to available memory for the OnLine system not the entire memory of the unix box.
- RA_PAGES - Set to 128.
- RA_THRESHOLD - Set to 120.
- DS_TOTAL_MEMORY - Set to 90% of SHMVIRTSIZE

13.1.4 Determining the port number of IDS on Linux

For a Java developer using WebSphere studio, one of the most important things to know is how to connect to the database server that interacts with the application. Typically you should contact the database administrator and get all the necessary information, such as:

- Host name
- IDS server (instance) name
- Database name
- TCP/IP port number
- User name and password with the necessary privileges

Most of this information has to be given by the database administrator, but you may be able to find some yourself. In this section we show how to find the port number configured for an IDS server instance.

Find the port number

Here are the steps to find the port number:

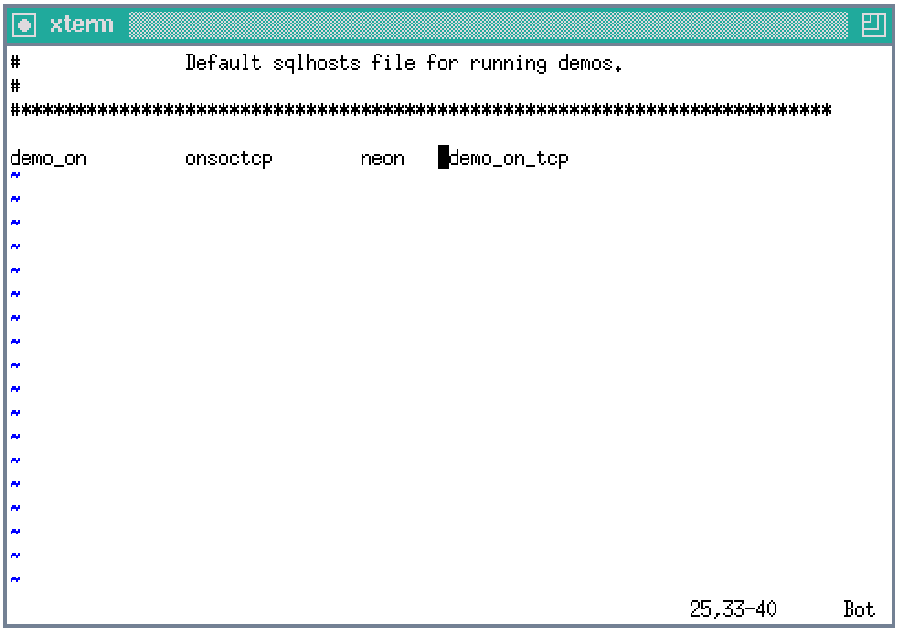
1. Log into the Linux server with the user id that was given by the database administrator (in our case, *itso*).
2. Open the sqlhosts file. You have to know the location of this file, since it can be in any directory, using the INFORMIXSQLHOSTS environment variable. You must ask your database administrator if this variable is set. If so, get the value stored and open the file:

   ```
   # vi $INFORMIXSQLHOSTS
   ```

   If INFORMIXSQLHOSTS is not configured the server uses the default file, so open it up using vi:

   ```
   # vi $INFORMIXDIR/etc/sqlhosts
   ```

3. Locate the correspondent entry of the instance name. The instance name is stored in the first column of the entry and the service name (port number) is stored on the forth column. In this case we have **demo_on** as the server name and **demo_on_tcp** as the service name. See Figure 13-7.



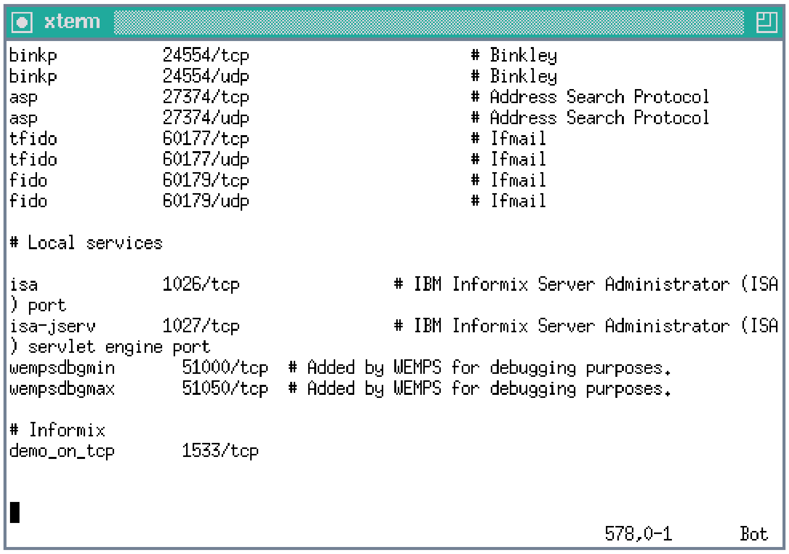
```
#                 Default sqlhosts file for running demos.
#
###################################################################################

demo_on          onsoctcp         neon    demo_on_tcp
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                 25,33-40       Bot
```

Figure 13-7 Sqlhosts file

4. Now that we know the service name we need to find out the TCP/IP port assigned to it. To do that, open the /etc/services file:

   # vi /etc/services

5. Search for service **demo_on_tcp** in the /etc/services file and find the port number. In our case the port **1533** is assigned to **demo_on_tcp**. Here are the port number and service name that have to be used in your application. See Figure 13-8.



```
binkp           24554/tcp                       # Binkley
binkp           24554/udp                       # Binkley
asp             27374/tcp                       # Address Search Protocol
asp             27374/udp                       # Address Search Protocol
tfido           60177/tcp                       # Ifmail
tfido           60177/udp                       # Ifmail
fido            60179/tcp                       # Ifmail
fido            60179/udp                       # Ifmail

# Local services

isa             1026/tcp                # IBM Informix Server Administrator (ISA
) port
isa-jserv       1027/tcp                # IBM Informix Server Administrator (ISA
) servlet engine port
wempsdbgmin       51000/tcp  # Added by WEMPS for debugging purposes.
wempsdbgmax       51050/tcp  # Added by WEMPS for debugging purposes.

# Informix
demo_on_tcp       1533/tcp


                                                            578,0-1       Bot
```

Figure 13-8 Services file

13.1.5 Using sequence objects rather than serial data type

In this redbook we have presented a sample application that uses JSPs, html pages, servlets, and both entity-driven and message-driven beans. The application is used to query, insert, and update data on IDS. Data insertion is performed through a entity bean, and, in our case, a container managed persistence bean. The way that data is inserted is through a primary key and the data insertion is performed on the customers table that has **customer_num** as a primary key. This column is defined as a serial data type, that caused us a problem when using the entity bean. For serial data types there are two ways to insert data:

- Insert value 0, so the system generates the next unique value for the column
- Specify an unique number in the column

The first option is the simplest one since many applications may be accessing the server and inserting data into the table. The problem with this is that the conversion of value 0 to the next unique integer number is performed on the database server. So after inserting the new value the entity bean has to query this row and update the other columns. This did not work for our application because WebSphere was looking for a primary key of value 0 instead of the unique number generated by IDS.

The solution for this problem is to use *sequence objects.* This database object was introduced with IDS 9.40 and is described in Chapter 2. A sequence object generates unique numbers on the server, so we used the second approach to insert serial values. First we create a sequence object using dbaccess:

```
dbaccess stores_demo
```

```
create sequence custnum start 150 increment 1
```

The command above creates a sequence object that starts with value 150 and increments it by 1.

Note: In our case we only used the stores_demo database for the sample applications so the first record inserted has a customer_num of 150.

Now that we have our sequence object we can determine the next unique value. In our application we run the following SQL statement before an insert:

```
select custnum.nextval from systables where tabid=1
```

This statement returns the unique value that we can specify during the insertion of a new customer record. By doing this we avoid the problem with the serial data type.

13.1.6 WebSphere and IDS

In our environment we only installed WebSphere Application Server on Linux, however we found a problem that prevented us from launching the installation script. Whenever we tried to use LaunchPad.sh we received an error about a library called libc.so.6:

error while loading shared libraries: libc.so.6: cannot open shared object file:
No such file or directory

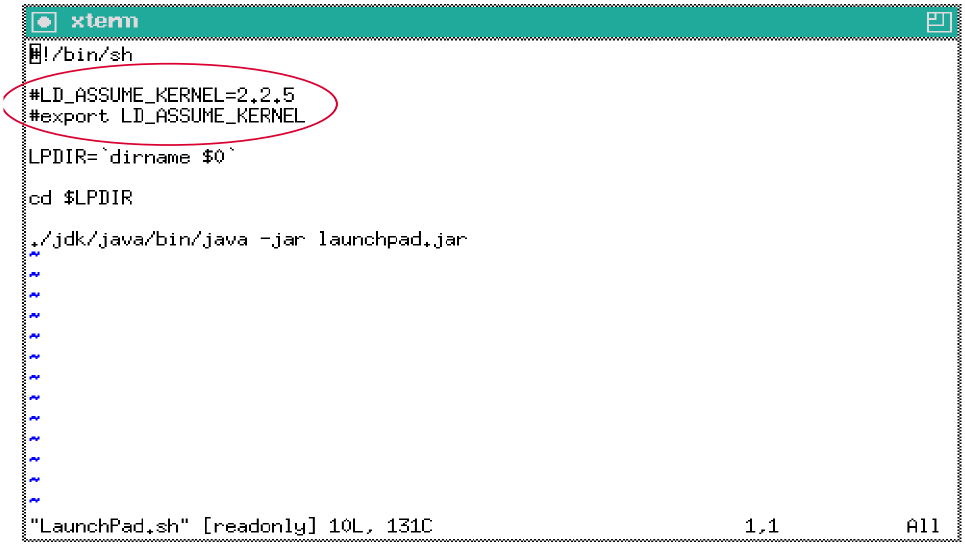
This is a known problem and is documented at:

http://sdb.suse.de/en/sdb/html/pthomas_install_anywhere.html

The solution is to edit the LaunchPad.sh script with the command:

```
vi LaunchPad.sh
```

The modified lines are highlighted in Figure 13-9:.



```
#!/bin/sh

#LD_ASSUME_KERNEL=2.2.5
#export LD_ASSUME_KERNEL

LPDIR=`dirname $0`

cd $LPDIR

./jdk/java/bin/java -jar launchpad.jar
~
~
~
~
~
~
~
~
~
~
~
~
~
"LaunchPad.sh" [readonly] 10L, 131C                        1,1            All
```

Figure 13-9 Editing LaunchPad.sh

After editing the script, the installation should work with no further problems.

13.1.7 Install error with Redhat 8.0 Linux

Prior to our installation and use of SuSE 8.0 Linux, we tried installing and configuring an Intel server with Redhat 8.0 Linux. We then installed IDS 9.40 and WAS Enterprise Edition 5.0. The installation of both products was easy and smooth, and no problems were encountered.

However when we began our testing process using WSAD, there was a problem with using IDS on this version of Linux. Here are some of the symptoms of the problem. When connections were established with the database server (usually remote connections, such as with WSAD on Windows) and database activities were performed, IDS went into a state called *checkpoint request*. Because of this we were unable to continue using RedHat Linux and moved to SuSE Linux Version 8. Everything worked without problems on this version of Linux.

13.1.8 An alternative Java UDR deployment method

In Section 9.3, “Using IDS 9 as a Web service consumer” on page 233, are two examples of Java UDRs that have to be registered with the IDS 9.40 database server by using an SQL script and a two step approach:

1. execute the install_jar() routine
2. execute a create function() statement.

In addition to this approach, there is an even easier way to achieve the same result and that is by utilizing a deployment descriptor in combination with a manifest file.

Let’s take a look at the necessary steps by using the example given in 9.3.5, “A simple IDS Web service example — Currency Exchange project” on page 239:

1. Create a deployment descriptor file with the name deploy.txt and the content shown in Example 13-1.

Example 13-1 deploy.txt file

```
SQLActions[] = {
   "BEGIN INSTALL

   create function CurrencyExchange(lvarchar, lvarchar)
   returns float as exchange_rate
external name 'thisjar:CurrencyExchangeUDRs.currencyExchange.(java.lang.String,
                      java.lang.String)'
   language java;

   alter function CurrencyExchange(lvarchar, lvarchar) with
                         (add parallelizable);

   grant execute on function CurrencyExchange(lvarchar, lvarchar)
                         to public;
   END INSTALL",

   "BEGIN REMOVE
   drop function CurrencyExchange(lvarchar, lvarchar);
   END REMOVE"
}
```

2. Create a manifest file with the name manifest.txt and the content shown in Example 13-2.

Example 13-2 manifest.txt file

```
Name: deploy.txt
SQLJDeploymentDescriptor: True
```

3. Finally, create the CurrencyExchange.jar file by executing the following command:

```
jar cvmf manifest.txt CurrencyExchange.jar deploy.txt CurrencyExchange\*.class
```

Now we have a self contained jar file which can be used in combination with an alternative install_jar() routine call.

To deploy the CurrencyExchange.jar file you should call install_jar() with the following options:

```
execute procedure install_jar( 'file:C:/RedBook2003/AXIS/CurrencyExchange.jar',
'CurrencyExchange', 1);
```

This command will install the jar file in the database and execute the SQL statements in the BEGIN INSTALL section of deployment descriptor file.

To drop the CurrencyExchange UDR and remove the CurrencyExchange.jar file, please call remove_jar() with the following options:

```
execute procedure remove_jar(’CurrencyExchange’, 1);
```

This command will remove the jar file in the database and execute the SQL statements in the BEGIN REMOVE section of deployment descriptor file.

Summary

By applying the deployment method above, the user does not have to re-create/drop the UDRs every time the jar file is installed/removed.

For more detailed information on this topic, please refer to the section, “Using a Deployment Descriptor”, in the *IBM Informix J/Foundation Developer’s Guide*[1].

[1] http://publibfi.boulder.ibm.com/epubs/pdf/ct1szna.pdf

A

SQLtoXML and XMLtoSQL Java class description

This appendix contains additional information about the SQLtoXML and the XMLtoSQL Java classes. We describe the SQLtoXML and XMLtoSQL classes in more detail, while the QueryProperties, SQLProperties and BaseProperties are done in an overview/table style.

The class libaries, their documentation and some examples are located at:

```
<WSAD5_Installdir>\wstools\eclipse\plugins\com.ibm.etools.sqltoxml_5.0.1.
```

Class com.ibm.etools.sqltoxml.SQLToXML

```
java.lang.Object com.ibm.etools.sqltoxml.SQLToXML
```

public class **SQLToXML**
extends java.lang.Object

This class provides methods used by applications (such as the SQL to XML wizard) to perform database queries. The query result is obtained in an XML format. Optionally, corresponding DTD, XML schema, and XSL files of the result can be generated.

Constructors

SQLToXML

```
public SQLToXML(QueryProperties qProperties)
```

This is the only constructor.

Methods

execute

```
public void execute() throws java.lang.Exception
```

Executes a query based on information from QueryProperties given as the constructor argument. The artifacts that can be generated include XML, XML schema, DTD, and XSL files. To generate artifacts other than XML, the file names or PrintWriters need to be provided by using relevant set accessors before using this method. Additionally the XML output can be obtained as a DOM document through the getCurrentDocument() or getCurrentDocuments() method call.

Throws: `java.lang.Exception` - Thrown when the execution fails.

execute

```
public void execute(PrintWriter xml, String dtdfile, String xsdfile, PrintWriter
xsl) throws Exception
```

Deprecated. Executes a query based on information from QueryProperties and the results are written to the PrintWriters and/or files. When any argument other than 'xml' is null, no result is generated for that particular argument. Also, if both dtdfile and xsdfile are specified, only xsdfile is generated.

Parameters

xml - A PrintWriter for an XML result.
dtdfile - A DTD file name for the xml.
xsdfile - An XML schema file name for the xml.
xsl - A PrintWriter for a default XSL.

Throws: Exception Thrown when the execution fails.

See Also: QueryProperties

execute

```
public void execute(String params, PrintWriter xml, String dtdfile, String xsdfile,
PrintWriter xsl) throws Exception
```

Deprecated. Executes a query based on information from QueryProperties and the results are written to the PrintWriters and/or files. When any argument other than 'xml' is null, no result is generated for that particular argument. Also, if both dtdfile and xsdfile are specified, only xsdfile is generated. This method is used when the query contains a where-clause and its constraints are given in 'params' String argument. These constraint parameters are delimited using comma (,) in the argument.

Parameters

params - where-clause constraint parameters delimited by comma(',').
xml - A PrintWriter for an XML result.
dtdfile - A DTD file name for the xml.
xsdfile - An XML schema file name for the xml.
xsl - A PrintWriter for a default XSL.

Throws: Exception Thrown when the execution fails.

See Also: QueryProperties

execute

```
public void execute(String xmlfile, String dtdfile, String xsdfile, String xslfile)
throws Exception
```

Deprecated. Executes a query based on information from QueryProperties and the results are written to the files. When any argument other than 'xml' is null, no result is generated for that particular argument. Also, if both dtdfile and xsdfile are specified, only xsdfile is generated.

Parameters

xmlfile - An XML file name for the query result.
dtdfile - A DTD file name for the xmlfile.
xsdfile - An XML schema file name for the xmlfile.
xslfile - An XSL file name for a default XSL.

Throws: Exception Thrown when the execution fails.

See Also: QueryProperties

finalize

```
public void finalize()
```

Closes the connection if this was created by SQLToXML. If the connection was supplied externally (by using setConnection()), the connection is not closed. This method is called by the JVM when SQLToXML is garbage-collected.

getConnection

```
public Connection getConnection() throws Exception
```

Returns a database connection. If no connection exists, this method creates a new connection using the values from QueryProperties.

Returns: A JDBC connection used for the query

Throws: Exception Thrown when a connection is not available.

getCurrentDocument

```
public Document getCurrentDocument()
```

Returns the XML document that is a result of the most recent call to execute() method. If the GENERATE_ID_AND_IDREF with RECURSIVE option is used, the main XML document is returned.

Returns: An XML document

See Also: getCurrentDocuments

getCurrentDocuments

```
public Document[] getCurrentDocuments()
```

Returns an XML documents array that is a result of the most recent call to execute() method. Unless the GENERATE_ID_AND_IDREF with RECURSIVE option is used, the result contains only one document.

Returns: An array of XML documents.

See Also: getCurrentDocument

getDTDFile

```
public java.lang.String getDTDFile()
```

Returns current DTD file name to be generated.

Returns: A DTD file name.

See Also: setDTDFile

getNextAllQuery

```
public String getNextAllQuery()
```

Returns a valid query string. This method returns a valid query string only when

1) Format option is ID_AND_IDREF, and
2) Recurse option is 'TRUE'

For example, when a column from the current query is a foreign key to a primary key column of table A, this will return a string 'select * from A'. Refer to the sample servlet XMLIntegratorServlet.java for a typical use of this method.

Returns: A select statement for a table referenced by a foreign key

getNextQueries

```
public Vector getNextQueries()
```

Returns a sequence of valid query strings. This method returns a Vector of valid query strings only when

1) Format option is ID_AND_IDREF, and
2) Recurse option is 'TRUE'

For example, when a column from the current query is a foreign key to a primary key column ID (with value 123) of table A, this will return a string 'select * from A where ID = 123'. Refer to the sample servlet XMLIntegratorServlet.java for a typical use of this method.

Returns: A select statement (with where-clause) for a table row referenced by a foreign key

getParameters

```
public java.lang.String getParameters()
```

Returns the parameter values for the where-clause.

Returns: The where-clause parameter values when they exist.

See Also: setParameters

getXMLFile

```
public java.lang.String getXMLFile()
```

Returns current XML file name to be generated.

Returns: An XML file name.

See Also: setXMLFile

getXMLWriter

```
public java.io.PrintWriter getXMLWriter()
```

Returns current PrintWriter for the XML output.

Returns: A PrintWriter for the XML output.

See Also: setXMLWriter

getXSDFile

```
public java.lang.String getXSDFile()
```

Returns current XML schema file name to be generated.

Returns: An XML schema file name.

See Also: setXSDfile

getXSLFile

```
public java.lang.String getXSLFile()
```

Returns current XSL file name to be generated.

Returns: An XSL file name.

See Also: setXSLFile

getXSLWriter

```
public java.io.PrintWriter getXSLWriter()
```

Returns current PrintWriter for the XSL output.

Returns: A PrintWriter for the XSL output.

See Also: setXSLWriter

setConnection

```
public void setConnection(Connection jdbcConnection)
```

Provides a JDBC connection to be used for generating the XML files. This method becomes useful when an application program wants to use its own connection management mechanism such as connection pooling. If this method is not called before any of the execute() methods, SQLToXML creates its own connection using the values from QueryProperties.

Parameters

jdbcConnection - A JDBC connection

setDTDFile

```
public void setDTDFile(java.lang.String filename) throws java.lang.Exception
```

Sets the DTD file name to be generated. This method cannot be used in combination with setXSDFile(String).

Parameters:

filename - A DTD file name to be generated

Throws: java.lang.Exception - Thrown if setXSDFile(String) has been called already.

setGenDocType

```
public void setGenDocType(Boolean value)
```

Deprecated. Controls whether or not to generate a DTD or an XML schema file for validation purpose. By default, the value is set to true for development time, and to false for runtime. Setting the value to true for the runtime applications such as servlets will cause some performance decrease.

Parameters

value - false for not generating a validation file, and true otherwise.

setIndenting

```
public void setIndenting(Boolean flag)
```

Sets the indentation on and off. Indentation is on by default.

Parameters

flag - false if indentation should be off.

setMaxRows

```
public void setMaxRows(int max)
```

Sets the limit for the maximum number of rows to be retrieved in the generated XML to max. By default, there is no limit.

Parameters

max - The limit for the number of rows. Zero means there is no limit.

setParameters

```
public void setParameters(java.lang.String parameters)
```

Sets the parameter values for the where-clause. This method is intended to be used when the query contains the where-clause. For example,

select * from customer where customer_num = :custnum

From the above example, the value to be replaced for :custnum is given in the parameters argument. If there are multiple constraint values, they are delimited using comma (,) in the parameters like in the following example.

String params = "85737,'Ismaning'";

sqltoxml.setParameters(params);

Parameters:

parameters - The where-clause parameter values when they exist

setQueryProperties

```
public void setQueryProperties(QueryProperties qProperties)
```

Overrides current queryProperties.

setXMLFile

```
public void setXMLFile(java.lang.String filename) throws java.lang.Exception
```

Sets the XML file name to be generated. This method cannot be used in combination with setXMLWriter(PrintWriter).

Parameters:

filename - An XML file name to be generated

Throws: java.lang.Exception - Thrown if setXMLWriter(PrintWriter) or setXSLWriter(PrintWriter) has been called already.

See Also: setXMLWriter

setXMLWriter

```
public void setXMLWriter(java.io.PrintWriter writer) throws java.lang.Exception
```

Sets the PrintWriter to which the generated XML is directed. This method cannot be used in combination with setXMLFile(String).

Parameters:

filename - A PrintWriter where the XML is going to be directed.

Throws: java.lang.Exception - Thrown if setXMLFile(PrintWriter) or setXSLFile(PrintWriter) has been called already.

See Also: setXMLFile

setXSDFile

```
public void setXSDFile(java.lang.String filename) throws java.lang.Exception
```

Sets the XML schema file name to be generated. This method cannot be used in combination with setDTDFile(String).

Parameters:

filename - An XML schema file name to be generated

Throws: java.lang.Exception - Thrown if setDTDFile(String) has been called already.

setXSLFile

```
public void setXSLFile(java.lang.String filename)throws java.lang.Exception
```

Sets the XSL file name to be generated. This method cannot be used in combination with setXSLWriter(PrintWriter).

Parameters:

filename - An XSL file name to be generated

Throws: java.lang.Exception - Thrown if setXSLWriter(PrintWriter) or setXMLWriter(PrintWriter) has been called already.

See Also: setXSLWriter

setXSLWriter

```
public void setXSLWriter(java.io.PrintWriter writer) throws java.lang.Exception
```

Sets the PrintWriter to which the generated XSL is directed. This method cannot be used in combination with setXSLFile(String).

Parameters:

filename - A PrintWriter where the XSL is going to be directed.

Throws: java.lang.Exception - Thrown if setXSLFile(PrintWriter) or setXMLFile(PrintWriter) has been called already.

See Also: setXSLWriter

Class com.ibm.etools.xmltosql.XMLToSQL

```
java.lang.Object com.ibm.etools.xmltosql.XMLToSQL
```

public class **XMLToSQL**
extends Object

XMLToSQL is used to insert, update, or delete rows in a database table using an XML document. The mapping between the XML structure to the table structure is based on a set of simple mapping rules. The XML fragment shown in Example A-1 illustrates the mapping rules:

Example: A-1 Mapping rules

```
<rootElement>
    <customer>
<fname>Omkar</fname>
      <lname>Nimbalkar</lname>
    </customer>
    <customer>
      <fname>Chuck</fname>
      <lname>Ballard</lname>
    </customer>
    ...
  </rootElement>
```

The root element can contain 0-n elements. The tag of the root element is irrelevant. All elements contained by the root will be processed based on the specified action type.

Each element maps to a row in the corresponding table. In this example, the customer element maps to the customer table. The children of the customer element maps to the 2 columns in the customer table by name. That is, the fname element corresponds to the fname column in the customer table, the lname element corresponds to the lname column in the customer table.

The XMLToSQL library will create the appropriate SQL statement (e.g. insert, update, or delete) based on the column value and the data type for the corresponding column in the table.

A good way to create a valid XML document is to first use the SQLToXML library to generate one, then modify it to supply new values for update. You can also use the XML to SQL wizard to unit test the XML document for validity before writing your code that will use the XMLToSQL library.

Constructors

XMLToSQL

```
public XMLToSQL(SQLProperties sqlProperties)
```

This is the only constructor. Any information necessary for updating a database table is provided through SQLProperties.

Parameters

sqlProperties - Contains information for the update.

See Also: SQLProperties

Methods

addToKeyColumns

```
public void addToKeyColumns(String columnName) throws Exception
```

Adds to the list of the key column names that will be used in building the where-clause for UPDATE or DELETE. If the table has primary keys, those primary key columns are used instead. This method is useful if a table does not have any primary key columns defined, but requires caution in this case since multiple rows can be updated unintentionally. A column cannot be both a key column and an update column. This method does not have any effect for INSERT.

Parameters

columnName - A column name to be used in the where-clause for UPDATE or DELETE.

Throws: Exception The column in the update column list cannot be used as a key column.

See Also: addToUpdateColumns

addToUpdateColumns

```
public void addToUpdateColumns(String columnName) throws Exception
```

Adds to the list of the column names that will be updated or inserted. If this method is not called before any of the execute() methods, all of the columns excluding any key columns are added to the list by XMLToSQL for UPDATE or INSERT. A column cannot be both a key column and an update column. This method does not have any effect for DELETE..

Parameters

columnName - A column name to be updated or inserted in the table.

Throws: Exception The column in the key column list cannot be used as an update column.

See Also: addToKeyColumns

execute

```
public void execute(Document doc,Boolean continueOnSQLError) throws SQLException,
ClassNotFoundException, IOException, SAXException, ParserConfigurationException
```

Updates a table from a DOM Document. Different transaction mode is applied depending on the Boolean value of continueOnSQLError. When this is true, updating continues for the rest of the rows even if an error occurs. To see which rows have been failed in this case, you need to call getFailedStatements(). If continueOnSQLErrors is false, rollback is done on all of the rows processed should an error occurs.

Parameters

doc - DOM Document.
continueOnSQLError - Used to choose a transaction mode.

Throws

SQLException DDL execution failed.
ClassNotFoundException JDBC driver could not be found.
IOException Corrupted XML stream
SAXException The XML content of the input stream may not be well-formed.
ParserConfigurationException Could not find an XML parser.

See Also: getFailedStatements

execute

```
public void execute(InputStream inputStream) throws SQLException,
ClassNotFoundException, IOException, SAXException, ParserConfigurationException
```

Updates a table from an XML input stream. If an error occurs while updating a row, rollback is performed for all of the rows processed so far.

Parameters

inputStream - XML input stream.

Throws

SQLException DDL execution failed.
ClassNotFoundException JDBC driver could not be found.
IOException Corrupted XML stream
SAXException The XML content of the input stream may not be well-formed.
ParserConfigurationException Could not find an XML parser.

execute

```
public void execute(InputStream inputStream,Boolean continueOnSQLError) throws
SQLException, ClassNotFoundException, IOException, SAXException,
ParserConfigurationException
```

Updates a table from an XML input stream. Different transaction mode is applied depending on the Boolean value of continueOnSQLError. When this is true, updating continues for the rest of the rows even if an error occurs. To see which rows have been failed in this case, you need to call getFailedStatements(). If continueOnSQLErros is false, rollback is done on all of the rows processed should an error occurs.

Parameters

inputStream - XML input stream.
continueOnSQLError - Used to choose a transaction mode.

Throws

SQLException DDL execution failed.
ClassNotFoundException JDBC driver could not be found.
IOException Corrupted XML stream
SAXException The XML content of the input stream may not be well-formed.
ParserConfigurationException Could not find an XML parser.

See Also: getFailedStatements

execute

```
public void execute(String filename) throws SQLException, FileNotFoundException,
ClassNotFoundException, IOException, SAXException, ParserConfigurationException
```

Updates a table from an XML file. If an error occurs while updating a row, rollback is performed for all the rows processed so far.

Parameters

filename - XML filename.

Throws

SQLException DDL execution failed.
FileNotFoundException Input XML file was not found.
ClassNotFoundException JDBC driver could not be found.
IOException Corrupted XML file.
SAXException The XML content of the input file may not be well-formed.
ParserConfigurationException Could not find an XML parser.

execute

```
public void execute(String filename,Boolean continueOnSQLError) throws
SQLException, FileNotFoundException, ClassNotFoundException, IOException,
SAXException, ParserConfigurationException
```

Updates a table from an XML file. Different transaction mode is applied depending on the Boolean value of continueOnSQLError. When this is true, updating continues for the rest of the rows even if an error occurs. To see which rows have been failed in this case, you need to call getFailedStatements(). If continueOnSQLErros is false, rollback is done on all of the rows processed should an error occurs.

Parameters

filename - XML filename.
continueOnSQLError - Used to choose a transaction mode.

Throws

SQLException DDL execution failed.
FileNotFoundException Input XML file was not found.
ClassNotFoundException JDBC driver could not be found.
IOException Corrupted XML file
SAXException The XML content of the input file may not be well-formed.
ParserConfigurationException Could not find an XML parser.

See Also: getFailedStatements

finalize

```
public void finalize()
```

Closes the connection if this was created internally. If the connection was supplied externally (by using setConnection()), nothing is done. This method is called by the JVM when XMLToSQL is garbage-collected.

Overrides: finalize in class Object

See Also: setConnection

getFailedStatements

```
public Vector getFailedStatements()
```

Returns a collection of SQL statements that have been failed during execution. This method is only meaningful when the continueOnError flag is true.

Returns: A (String) Vector of SQL statements that have failed to execute.

getTableName

```
public String getTableName(InputStream inputStream) throws IOException,
SAXException, ParserConfigurationException
```

Returns the database table name to be updated. This method is provided to determine the table name from the XML input stream before executing a SQL statement. This is for the convenience of any user interface components.

Parameters

inputStream - XML input stream

Returns: A table name retrieved from the XML stream.

Throws

IOException Corrupted XML stream
SAXException The XML content of the input stream may not be well-formed.
ParserConfigurationException Could not find an XML parser.

getTableName

```
public String getTableName(String filename) throws IOException, SAXException,
ParserConfigurationException
```

Returns the database table name to be updated. This method is provided to determine the table name from the XML input stream before executing a SQL statement. This is for the convenience of any user interface components.

Parameters

filename - XML file name

Returns: A table name retrieved from the XML file.

Throws

IOException Corrupted XML stream
SAXException The XML content of the input stream may not be well-formed.
ParserConfigurationException Could not find an XML parser.

setConnection

```
public void setConnection(Connection jdbcConnection)
```

Provides a JDBC connection to be used for updating the tables. This method becomes useful when an application program wants to use its own connection management mechanism such as connection pooling. If this method is not called before any of the execute() methods, XMLToSQL creates its own connection using the values from SQLProperties.

Parameters

jdbcConnection - A JDBC connection

setTrace

```
public void setTrace(Boolean flag)
```

Sets the trace flag. The SQL statements being executed is printed out to the console when flag is true. By default, trace is turned off.

Parameters

flag - true to show the trace, false otherwise.

setTrace

```
public void setTrace(Boolean flag, PrintWriter writer)
```

Sets the trace flag. The SQL statements being executed is printed out to the writer when flag is true. By default, trace is turned off.

Parameters

flag - true to show the trace, false otherwise.

writer - A PrintWriter where the trace output is directed to.

updateMultipleRows

```
public Boolean updateMultipleRows(String uri, Vector keys) throws Exception
```

Determines if multiple rows would be updated/deleted once any of the execute() method was called. There is a chance of multiple rows being changed in UPDATE/DELETE mode when the operation is performed against a table that does not have any primary keys. This method returns false if a single row is going to be affected from its corresponding source XML element, and true otherwise.

Parameters

uri - A source XML filename

keys - The key column names to be used in the where-clause for UPDATE/DELETE

Returns: false for single row update, true otherwise.

Throws: Exception Failed while processing the source XML file.

Class com.ibm.etools.sqltoxml.QueryProperties

```
java.lang.Object com.ibm.etools.sqltoxml.BaseProperties |
com.ibm.etools.sqltoxml.QueryProperties
```

public final class **QueryProperties**
extends BaseProperties

This class provides information necessary for SQLToXML to perform its SQL query and generate an XML file as well as several other artifacts. The information can be either set manually or loaded from a query file (which usually has an extension .xst). Also, the information can be stored to a query file.

Constructor overview

Constructor	Description
QueryProperties()	The default constructor.

Method overview

Method	Description
String getEncoding()	Returns the Java style encoding value for the properties file.
String getEncodingTag()	Returns the encoding value for the properties file to be written if store() is called.
String getFormat()	Returns a format option.
String getJdbcDriver()	Returns a JDBC driver name.
String getJdbcServer()	Returns a JDBC server path.
String getLoginId()	Returns a database user ID that has the appropriate privilege to perform the query.
String getPassword()	Returns a password for the user ID that has the appropriate privilege to perform the query.
Boolean getRecurse()	Returns a Boolean value indicating whether or not the queries should also be performed on the target tables when foreign key columns from the current query are found.
String getStatement()	Returns current SQL statement to be executed.
Vector getVarTypes() Deprecated.	Returns a sequence of substitution parameter types when used.
void load(InputStream)	Loads necessary query values from an InputStream.
void load(String)	Loads necessary query values from a query file.
void setEncoding(String)	Sets the Java style encoding value for the properties file to be written if store() is called.

Method	Description
void setEncodingTag(String)	Sets the encoding value for the properties file to be written if store() is called.
void setFormat(String)	Sets a format option.
void setJdbcDriver(String)	Sets a JDBC driver name.
void setJdbcServer(String)	Sets a JDBC server path.
void setLoginId(String)	Sets a database user ID that has a privilege to perform the query.
void setPassword(String)	Sets a password for the user ID that has a privilege to perform the query.
void setRecurse(Boolean)	Sets a Boolean value indicating whether or not the queries should also be performed on the target tables when foreign key columns are found from the current query.
void setStatement(String)	Sets current SQL statement.
void setVarTypes(Vector) Deprecated.	VarTypes are used when a SQL statement contains substitution parameters, such as '?', and the user wants to indicate the data types for those parameters explicitly.
void store(String)	Stores necessary query values to a query file.

Class com.ibm.etools.xmltosql.SQLProperties

```
java.lang.Object com.ibm.etools.sqltoxml.BaseProperties
com.ibm.etools.xmltosql.SQLProperties
```

public final class **SQLProperties**
extends BaseProperties

This class is used to provide data necessary for XMLToSQL to update database tables. An instance of this class is used as an argument to the XMLToSQL constructor.

Field overview

Field	Description
DELETE	SQL DELETE action type.
INSERT	SQL INSERT action type.
UPDATE	SQL UPDATE action type.

Constructor overview

Constructor	Description
SQLProperties()	The default constructor.

Method overview

Method	Description
String getAction()	Returns the action to be performed by XMLToSQL.
String getSchema()	Returns the schema name.
void load(InputStream)	Loads necessary database manipulation values from an InputStream.
void load(String	Loads necessary database manipulation values from an external file.
void setAction(String)	Sets the action to be performed by XMLToSQL.
void setSchema(String)	Sets the schema name if required.
void store(String)	Stores database manipulation values to an external file.

Class com.ibm.etools.sqltoxml.BaseProperties

```
java.lang.Object com.ibm.etools.sqltoxml.BaseProperties
```

Direct Known Subclasses:

QueryProperties, SQLProperties

public abstract class **BaseProperties**
extends java.lang.Object

This class provides database connection information for SQL and XML. The database connection information can be either set manually or loaded from an external file (which usually has the extension .xst). The database connection information can also be stored in an external file with the extension .xst.

Constructor overview

Constructor	Description
BaseProperties()	The default constructor.

Method overview

Method	Description
String getEncoding()	Returns the Java style encoding value for the properties file.
String getEncodingTag()	Returns the encoding value for the properties file to be written if store() is called.
String getJdbcDriver()	Returns a JDBC driver name.
String getJdbcServer()	Returns a JDBC server path.
String getLoginId()	Returns a database user ID that has the appropriate privilege to perform the query.
String getPassword()	Returns a password for the user ID that has the appropriate privilege to perform the query.
void load(InputStream)	Loads database operation values from an InputStream.
void load(String)	Loads database operation values from an external file.
void setEncoding(String)	Sets the Java style encoding value for the properties file to be written if store() is called.
void setEncodingTag(String)	Sets the encoding value for the properties file to be written if store() is called.
void setJdbcDriver(String)	Sets a JDBC driver name.
void setJdbcServer(String)	Sets a JDBC server path.
void setLoginId(String)	Sets a database user ID that has a privilege to perform the query.
void setPassword(String)	Sets a password for the user ID that has a privilege to perform the query.
void store(String)	Stores database operation information to an external file.

B

DADX file format

This appendix includes the complete DADX syntax used in the Web Services scenario.

The DADX file is an XML document. The elements of the DADX are described here.

1. Root element: <DADX>

Attributes:

xmlns:dadx

The namespace of the DADX.

xmlns:xsd

The namespace of the W3C XML Schema specification

xmlns:wsdl

The namespace of the W3C Web services Definition Language specification

Children:

1.1 <wsdl:documentation>

Specifies a comment or statment about the purpose and content of the Web service. You can use XHTML tags.

1.2 <implements>

Specifies the namespace and location of the Web service description files. It allows the service implementor to declare that the DADX Web service implements a standard Web service described by a reusable WSDL document defined elsewhere; for example, in a UDDI registry.

1.3 <result_set_metadata>

Stored procedures can return one or more result sets which can be included in the output message. Metadata for a stored procedure result set must be defined explicitly in the DADX using the <result_set_metadata> element. At run-time, the metadata of the result set is obtained and it must match the definition contained in the DADX file.

Note: Therefore, only stored procedures that have result sets with fixed metadata can be invoked.

This restriction is necessary in order to have a well-defined WSDL file for the Web service. A single result set metadata definition can be referenced by several <call> operations, using the <result_set> element. The result set metadata definitions are global to the DADX and must precede all of the operation definition elements.

Attributes:

name Identifies the root element for the result set.

rowname Used as the element name for each row of the ressult set.

Children:

1.3.1 <column>

Defines the column. The order of the columns must match that of the result set returned by the stored procedure. Each column has a name, type and nullability, which must match the result set.

Attributes:

name Required. Specifies the name of the column.

type Required if element is not specified. Specifies the type column.

element Required if type is not specified. Specifies the element of column

as Optional. Provides a name for a columns.

nullable Optional. Nullable is either true or false. It indicates whether column values can be null.

1.4 <operation>

Specifies a Web service operation. The operation element and its children specify the name fo the operation, and the type of operation the Web service will perform, such as compose an XML document, query the database, or call a stored procedure. A single DADX file can contain multiple operations on a single database or location. The following list describes these elements.

Attribute:

name A unique string that identifies the operation; the string must be unique within the DADX file. For example: “findByCustomerID”

Children:

Document the operation with the following element:

1.4.1 <wsdl:documentation>

Specifies a comment or statement about the purpose and content of the operation. You can use XHTML tags.

Specify the type of operation using one of the following child elements:

1.4.2 <retrieveXML>

DB2 only! Specifies to generate zero or one XML documents from a set of relational tables using the XML collection access method. Depending on whether a DAD file or an XML collection name is specified, the opertation wil call the appropriate XML Extender composition stored procedure

Children:

Specify which of these stored procedures is used by passing either the name of a DAD file, or the name of the collection using one of the following elements:

1.4.2.1 <DAD_ref>

The content of this element is the name and path of a DAD file. If a relative path is specified for the DAD file, the current working directory is assumed to be the group directory.

1.4.2.2 <collection_name>

The content of this element is the name of the XML collection. Collections are defined using the XML Extender administration interfaces, as described in DB2 XML Extender Administration and Programming.

Specify override values with one of the following elements:

1.4.2.3 <no_override/>

Specifiies that the values in the DAD file are not overriden. Required if neither <SQL_override> nor <XML_override> element. Specifies a parameter for an operation. Use a separate parameter element for each parameter referenced in the operation. Each parameter name must be unique within the operation.

Attributes:

name The unique name of the parameter. A parameter must have its contents defined by either an XML Schema element (a complex type) or a simple type.

element Use the “element” attribute to specify an XML Schema element.

type Use the “type” attribute to specify a simple type.

kind Specifies whether a parameter passes input data, returns output data, or does both. The valid values for this attribute are:

- in

1.4.3 <storeXML>

DB2 only! Specifies to store (decompose) an XML dcoument in a set of relational tables using the XML collection access method. Depending on whether a DAD file or an XML collection name is specified, the operation will call the appropriate XML Extender decomposition stored procedure.

Children:

Specify which of these stored procedures is used by passing either the name of a DAD file, or the name of the collection using one fo the following elements:

1.4.3.1 <DAD_ref>

The content of this element is the name and path of a DAD file. If a relative path is specified for the DAD file, the current working directory is assumed to be the group directory.

1.4.3.2 <collection name>

The content of this element is the name of an XML collection. Collections are defined using the XML Extender administration interfaces, as described in DB2 XML Extender Administration and programming.

1.4.4 <query>

Specifies a query operation. The operation is defined by an SQL SELECT statement in the SQL_select> element. The statement can have zero or more named input parameters. If the statement has input parameters then each parameter is descibed by a <parameter> element.

This operation maps each database column from result set to a corresponding XML element. You can specify XML Extender user-defined types (UDTs) in the <query> operation, however this requires an <XML_result> element and a supporting DTD that defines the type of the XML column queried.

Children:

1.4.4.1 <SQL_query>

Sepcifies an SQL SELECT statement.

1.4.4.2 <XML_result>

Optional. Defines a named column that contains XML documents. The document type must be defined by the XML Schema element of its root.

Attributes:

name Specifies the root element of the XML document stored in the column.

element Specifies the particular element within the column.

1.4.4.3 <parameter>

Required when referencing a parameter in the <SQL_query> element. Specifies a parameter for an operation. Use a separate parameter element for each parameter referenced in the operation. Each parameter name must be unique within the operation.

Attributes:

name The unique name of the parameter. A parameter must have it contents defined by one of the following: an XML Schema element (a complex type) or a simple type.

element Use the “element” attribute to specify an XML Schema element.

type Use the “type” attribute to specify a simple type.

kind Specifies whether a parameter passes input data, returns output data, or does both. The valid values for this attribute are:

- in

1.4.5 <update>

The operation is defined by an SQL INSERT, DELETE, or UPDATE statement in the <SQL_update> element. The statement can have zero or more named input parameters. If the statement has input parameters then each parameter is described by a <parameter> element.

Children:

1.4.5.1 <SQL_update>

Specifies an SQL INSERT, UPDATE, or DELETE statement.

1.4.5.2 <parameter>

Required when referencing a parameter in the <SQL_update> element. Specifies a parameter for an operation. Use a separate parameter element for each parameter referenced in the operation. Each parameter name must be unique with the operation.

Attributes:

name The unique name of the parameter. A parameter must have its contents defined by one of the following: an XML Schema element (a complex type) or a simple type.

element Use the “element” attribute to specify an XML Schema element.

type Use the “type” attribute to specify a simple type.

kind Specifies whether a parameter passes input data, returns output data, or does both. The valid values for this attributes are:

- in

1.4.6 <call>

Specifies a call to a stored procedure. The processing is similar to the update operation, but the parameters for the call operation can be defined as ‘in’, ‘out’, or ‘in/out’. The defualt parameter kind is ‘in’. The ‘out’ and ‘in/out’ parameters appear in the output message.

1.4.6.1 <SQL_call>

Specifies a stored procedure call.

1.4.6.2 <parameter>

Required when referencing a parameter in the <SQL_call> element. Specifies a parameter for an operation. Use a separate parameter element for each parameter referenced in the operation. Each parameter name must be unique with the operation.

Attributes:

name The unique name of the parameter. A parameter must have its contents defined by one of the following: an XML Schema element (a complex type) or a simple type.

element Use the “element” attribute to specify an XML Schema element.

type Use the “type” attribute to specify a simple type.

kind Specifies whether a parameter passes input data, returns output data, or does both. The valid values for this attributes are:

- in
- out
- in/out

1.4.6.3 <result_set>

Defines a result set and must follow any <parameter> elements. The result set element has a name which must be unique among all teh parameters and result sets of the operation, and must refer to a <result_set_metadata> element. One <result_set> element must be defined for each result set returned from the stored procedure.

Attributes:

name A unique identifier for the result sets in the SOAP response.

metadata A result set meta data definition in the DADX file. The identifier must refer to the name of an element.

C

IDS and WSAD on Windows

Throughout this book, our installation has consisted of a Linux server and some Windows processors. We developed our sample applications using WebSphere Studio on the Windows processors, and deployed them on the Linux server that had both IDS and WebSphere Application Server installed.

In some cases, such as tests, demos, and workshops, it is interesting to have WebSphere products installed with IDS on a single server. To avoid the need to have lots of hardware resources, you can just install WSAD and IDS on a single server. WSAD has an embedded application server, and HTTP server that can be used in these situations.

This appendix provides an overview of the installation and environment configuration of IDS on a Windows processor (WSAD installation on Windows is already described in Chapter 4). Even though the installation processes are very similar to those for Linux (described in Chapter 2), there are some differences that are highlighted here. We also discuss some of the differences between IDS for Windows and IDS for Linux/UNIX.

Configuring IDS and WSAD for Windows

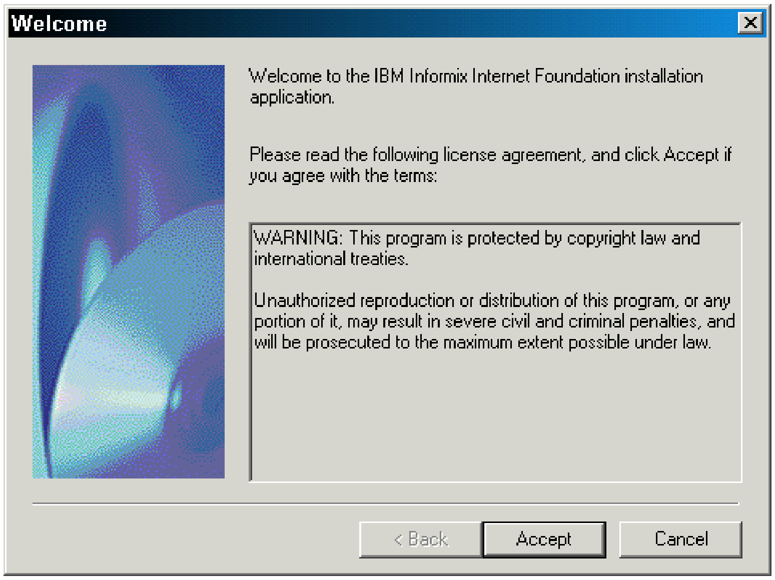
The installation of IDS on Windows is very similar to the process presented in Chapter 2. Again we need to consider whether you received IDS on a CD-ROM, or a compressed file.

1. If the product is on a CD-ROM: Just place the media in the CD-ROM drive and look for the *setup* application file (usually in the root of the CD hierarchy). Double click on this file and the installation process begins.
2. If the product is in a compressed file: Create a temporary directory to extract (WinZip) the product from the compressed file. Double-click on the *setup* application file and the installation process begins.

The following is the installation procedure for IDS. Screenshots, from our installation, are used for clarification on the important steps. We also discuss the installation of other products that support this environment, such as the JDK and Informix JDBC driver.

IDS installation steps

1. The first window that is displayed is a welcome to the installation wizard and a disclaimer is showed. Read the terms and click the **Accept** button.



Welcome to the IBM Informix Internet Foundation installation application.

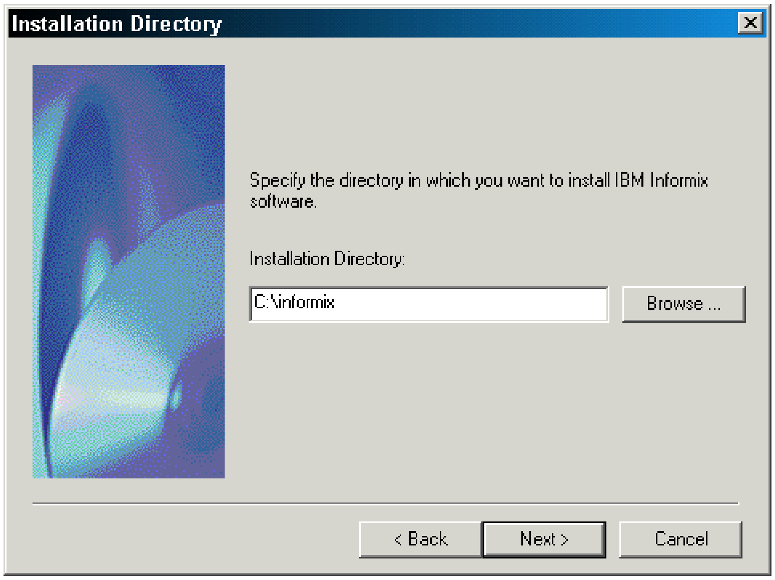
Please read the following license agreement, and click Accept if you agree with the terms:

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Figure C-1 Welcome window

2. Now we need to choose the installation directory for IDS. The recommendation is to install IDS on <DRIVE>:\Informix. Click **Next**.



Installation Directory

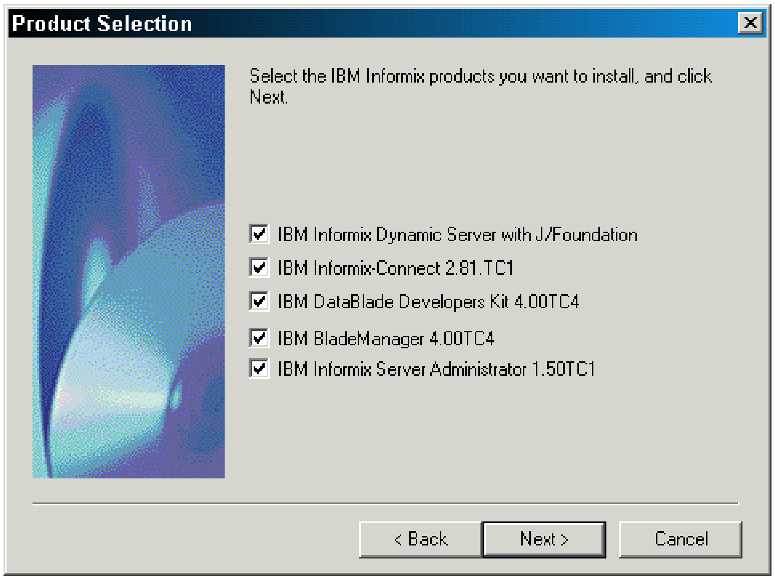
Specify the directory in which you want to install IBM Informix software.

Installation Directory:

C:\informix

Figure C-2 Installation directory

3. The next screen displays the products that come bundled with Informix Internet Foundation 9.40. In this section we only describe the installation of IDS, but you can install all the products available at this time if desired. They are installed automatically after IDS and no interaction is needed. In our case we selected all products. Select the ones that you want installed and click **Next**.



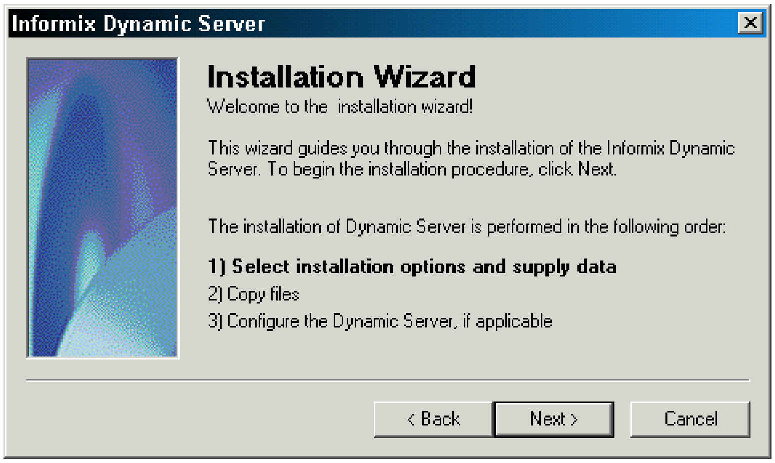
Product Selection

Select the IBM Informix products you want to install, and click Next.

☑ IBM Informix Dynamic Server with J/Foundation
☑ IBM Informix-Connect 2.81.TC1
☑ IBM DataBlade Developers Kit 4.00TC4
☑ IBM BladeManager 4.00TC4
☑ IBM Informix Server Administrator 1.50TC1

Figure C-3 Products available

4. The next window shows an overview of the installation process (Installation options, copy files, configure IDS). Click **Next**.


Informix Dynamic Server

Installation Wizard

Welcome to the installation wizard!

This wizard guides you through the installation of the Informix Dynamic Server. To begin the installation procedure, click Next.

The installation of Dynamic Server is performed in the following order:

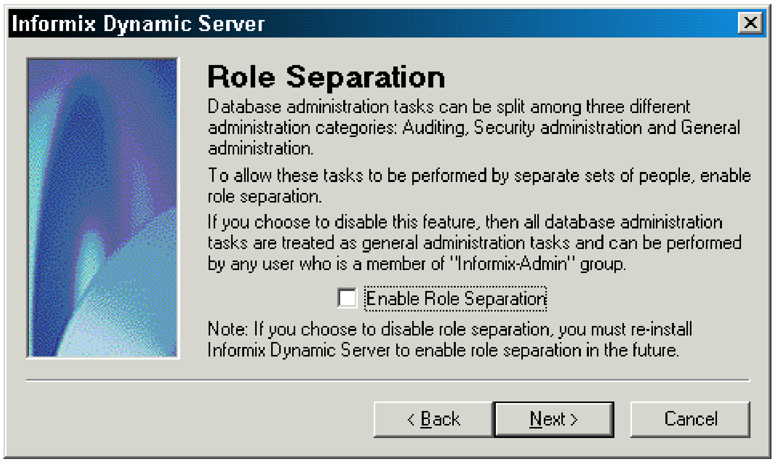
1) Select installation options and supply data

2) Copy files

3) Configure the Dynamic Server, if applicable

Figure C-4 Installation overview

5. Next the screen shows the option to perform a Windows domain installation. However, you need domain privileges to do that. We do not need this type of installation, so just click **Next**.
6. Now you have to choose the type of installation. Choose **Typical** and click **Next**.
7. The next window shows the option for Role Separation. Roles are used to split the administrative tasks of IDS among different groups and users. In our case we use user *informix* to do all that. Click **Next**.


Informix Dynamic Server

Role Separation

Database administration tasks can be split among three different administration categories: Auditing, Security administration and General administration.

To allow these tasks to be performed by separate sets of people, enable role separation.

If you choose to disable this feature, then all database administration tasks are treated as general administration tasks and can be performed by any user who is a member of "Informix-Admin" group.

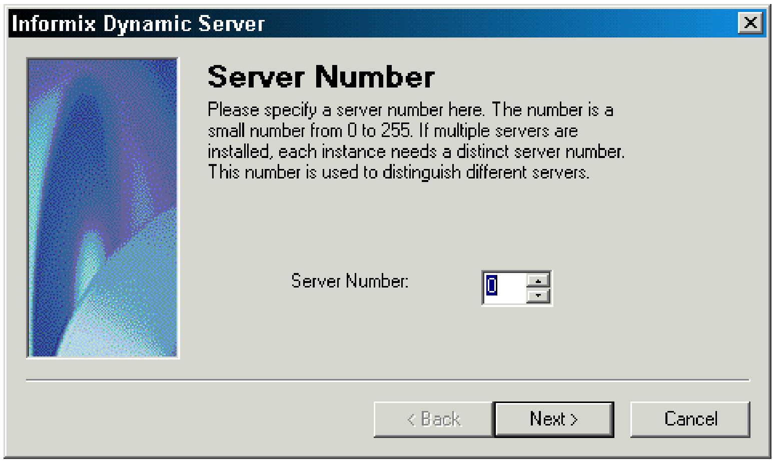
☐ Enable Role Separation

Note: If you choose to disable role separation, you must re-install Informix Dynamic Server to enable role separation in the future.

Figure C-5 Role separation

8. Now it is time to choose the password for our *informix* user. Here we also used *informix* as the password, but on a production system the password should be carefully chosen, since the *informix* user has privileges to do everything on the database server, just as the user *administrator* for the Windows System. Click **Next**.
9. The installation process is ready to copy the files into the directory chosen in Step 2. Just click **Next** and the copy begins.
10. After copying the files, it is time to configure a new server instance. On Linux some configuration is already pre-set, such as the name of the server and the server number. On Windows we have the choice to specify that.

11. Here we have to specify a number for our server instance (the first number is 0). Every new instance must have a unique server number. Since this is our first server configuration, we leave the number as `0`. Click **Next**.



Informix Dynamic Server

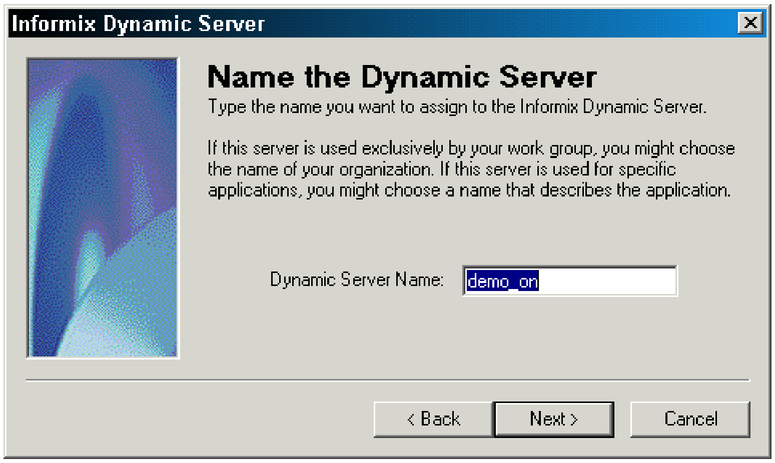
Server Number

Please specify a server number here. The number is a small number from 0 to 255. If multiple servers are installed, each instance needs a distinct server number. This number is used to distinguish different servers.

Server Number: 0

Figure C-6 Choosing server number

12. In the next window we have to choose a name for our server. To make the installation similar to the one for Linux, we also chose **`demo_on`**. Click **Next**.



Informix Dynamic Server

Name the Dynamic Server

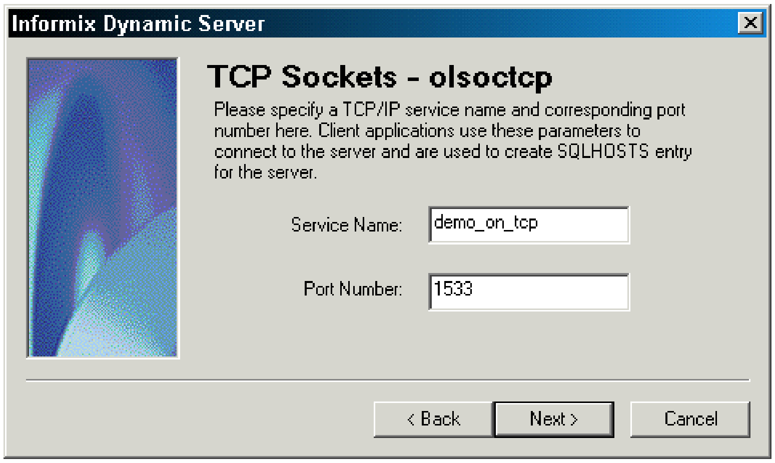
Type the name you want to assign to the Informix Dynamic Server.

If this server is used exclusively by your work group, you might choose the name of your organization. If this server is used for specific applications, you might choose a name that describes the application.

Dynamic Server Name: demo_on

Figure C-7 Choosing server name

13.On UNIX and Linux we have the option to specify a shared memory connection (onipcshm nettype on sqlhosts). On Windows the only way that clients applications connect to the server is through TCP/IP. In this window we have to choose a TCP/IP service name and number that will be configured for our client connections. We chose **`demo_on_tcp`** and **`1533`** to be consistent with the Linux installation (same port). You can choose a different port number as long it is unique in the services file. Click **Next**.



Informix Dynamic Server

TCP Sockets - olsoctcp

Please specify a TCP/IP service name and corresponding port number here. Client applications use these parameters to connect to the server and are used to create SQLHOSTS entry for the server.

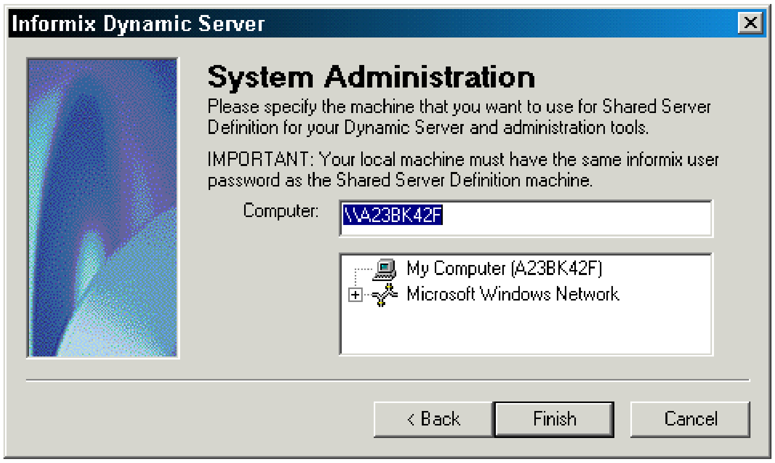
Service Name: demo_on_tcp

Port Number: 1533

Figure C-8 Specifying service name and TCP/IP port number

Tip: The services file on Windows is located in <WINNT>\system32\drivers\etc.

14.Now we have to choose the server to have our shared server definition (sqlhosts). Use the local server for that (the default). Click **Next**.



Informix Dynamic Server

System Administration

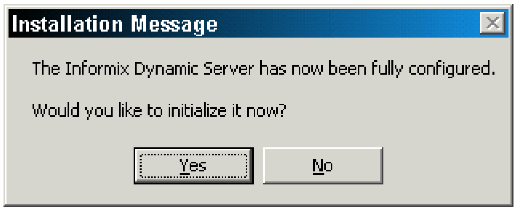
Please specify the machine that you want to use for Shared Server Definition for your Dynamic Server and administration tools.

IMPORTANT: Your local machine must have the same informix user password as the Shared Server Definition machine.

Computer: \\A23BK42F

Figure C-9 Shared server definition

15.The installation process updates the configuration files and the Windows registry, and prompts whether the configured server should be initialized. Click **Yes**. The server initialization takes a few minutes to complete.

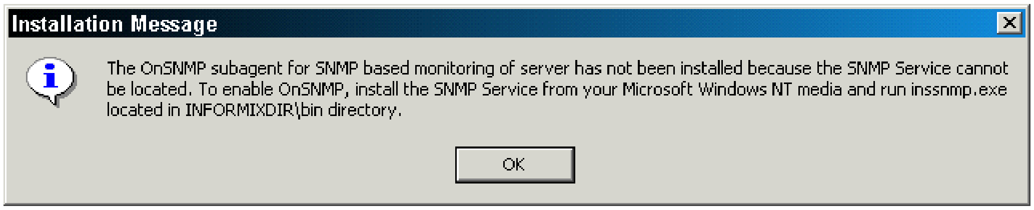

Installation Message

The Informix Dynamic Server has now been fully configured.

Would you like to initialize it now?

Figure C-10 Server ready to be initialized

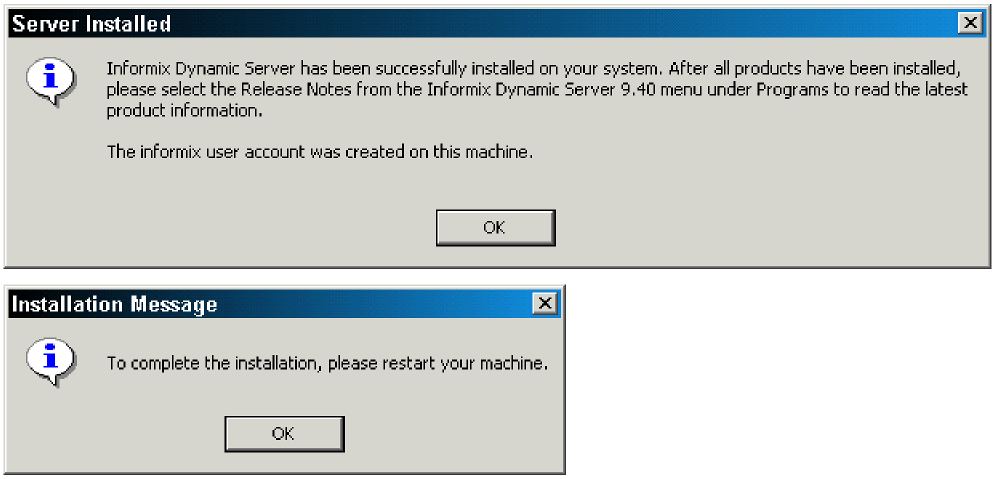
16. You might get a message saying that the OnSNMP subagent was not installed because the SNMP service was not present on the Windows system. OnSNMP is used to perform remote monitoring tasks. We do no use it so this message can be ignored. Just click **OK**.


Installation Message

The OnSNMP subagent for SNMP based monitoring of server has not been installed because the SNMP Service cannot be located. To enable OnSNMP, install the SNMP Service from your Microsoft Windows NT media and run inssnmp.exe located in INFORMIXDIR\bin directory.

Figure C-11 Message about OnSNMP

17. The installation of IDS is now complete. If you chose to install other products available in the bundle file they will be installed after IDS. We do not show the procedure here because the installation is automatic and you only need to hit the **Next** button. When everything is installed you will need to reboot the server to complete the installation procedure. Click **Yes** to reboot the server.


Server Installed

Informix Dynamic Server has been successfully installed on your system. After all products have been installed, please select the Release Notes from the Informix Dynamic Server 9.40 menu under Programs to read the latest product information.

The informix user account was created on this machine.

Installation Message

To complete the installation, please restart your machine.

Figure C-12 End of installation.

JDK and JDBC installation

The JDK and JDBC are both available for download:

JDK:

http://www.java.sun.com/j2se/downloads.html

Informix JDBC:

http://www-3.ibm.com/software/data/informix/

1. The installation of these two products is straightforward. First install the JDK and then configure its bin directory in the PATH environment variable. On a Windows/2000 you can do this by selecting **Start -> Settings -> Control Panel -> System -> Advanced -> Environment Variables**. Select the PATH variable and add the new setting. For example, after configuring the PATH environment variable path you should be able to see the JDK settings from a command window:

   ```
   C:\Eduardo\iif>echo %path%
   ```

   ```
   C:\WINNT\system32;C:\WINNT;C:\WINNT\system32\WBEM;c:\lotus\notes;C:\Program
   Files\Pcom;C:\Program Files\ObjREXX;C:\Program
   Files\ObjREXX\OODIALOG;c:\jdk1.3.1_07\bin
   ```

2. Now for the JDBC installation. Unzip the downloaded file with WinZip and double-click on the *setup.jar* file. Then follow the installation instructions. If you want to run Java applications on this server you must set the CLASSPATH environment variable. You have to include the <JDBC_DIR>\lib\ifxjdbc.jar and <JDBC_DIR>\lib\ifxjdbcx.jar files. You can do this following the same procedure that was used for the PATH variable in Step1. Again to verify that the settings are effective, run:

   ```
   C:\Eduardo\iif>echo %classpath%
   ```

   ```
   c:\Eduardo;c:\jdbc\lib\ifxjdbc.jar;c:\jdbc\lib\ifxjdbcx.jar;.
   ```

WSAD installation:

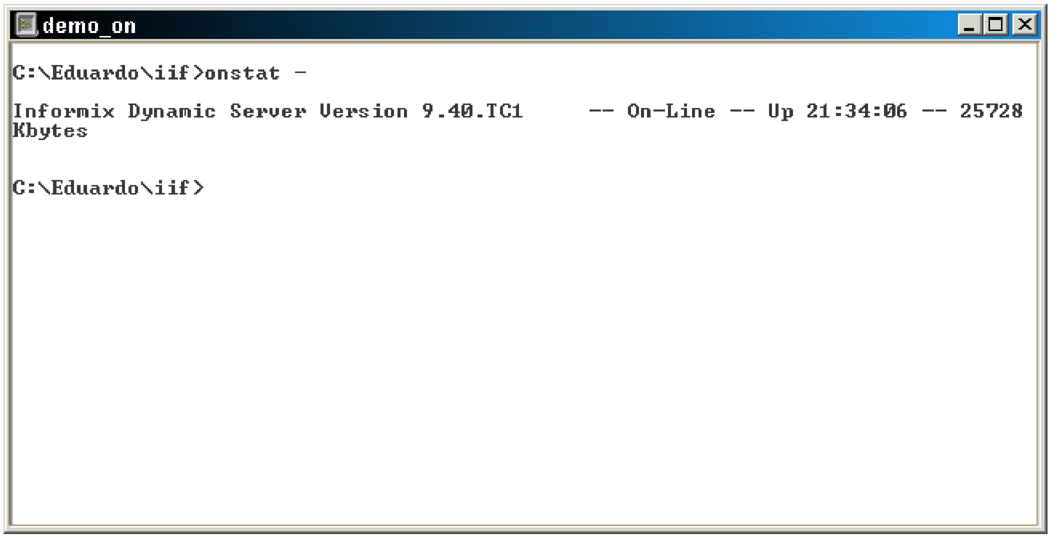
The installation of WSAD for Windows was described in detail in Chapter 4.

Windows/Linux Differences

The IDS functionality on Windows is basically the same as on Linux or UNIX, but there are some differences due to the nature of the Operating System. Some of these differences are highlighted here.

Accessing the server

The installation process gives you the option to create a first server instance (in our case, called *demo_on*). After the installation completes and the server is initialized you can access the server and run IDS commands through the command window. Notice that a new workgroup folder is created called Informix Dynamic Server 9.40. In this folder you see an icon in a form of a terminal with the name of your instance. Use this command-line window to run IDS commands since it pre-sets the correct environment variables for you (**Start -> Programs -> Informix Dynamic Server 9.40 -> demo_on**). You can do the same using a normal DOS command-line window but then you have to manually set these variables.



```
C:\Eduardo\iif>onstat -

Informix Dynamic Server Version 9.40.TC1     -- On-Line -- Up 21:34:06 -- 25728
Kbytes


C:\Eduardo\iif>
```

Figure C-13   IDS command-line window

Commands

IDS runs as a Windows service so there are different ways to start and stop the server.

Start the server:

1. Use the starts command:

   ```
   C:\Informix>starts demo_on
   ```

2. Use net command to start the IDS service:

   ```
   C:\Informix>net start demo_on
   ```

3. Use oninit. You can use this option too but the window will be frozen since the oninit command does not run in the background. You can still access IDS on a new window, but closing the one where the oninit command was issued will cause the server to stop.
4. Use Control panel -> Administrative tools -> Services to start the IDS service.

Stop the server:

1. Use the onmode command:

   ```
   C:\Informix>onmode -ky
   ```

2. Use the net command:

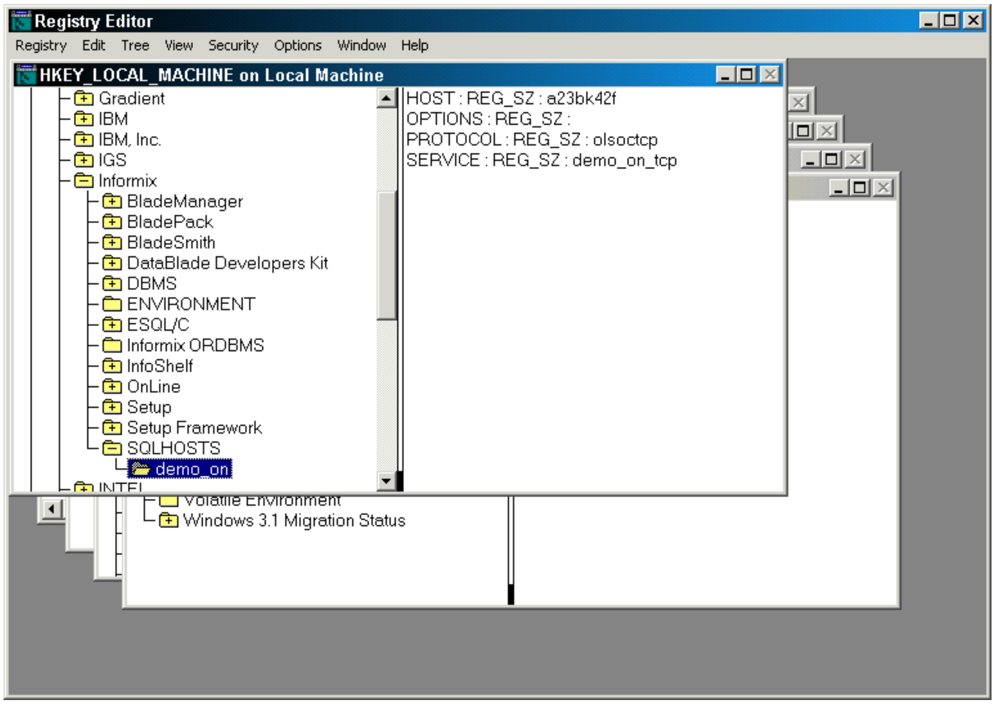
   ```
   C:\Informix>net stop demo_on
   ```

3. Use the Control Panel to stop the IDS service

SQLHOSTS on Windows

There is no *sqlhosts* file on Windows, the connection configuration is stored in the Windows registry. To add a new connection entry in sqlhosts you can either edit the register manually or install Informix Connect and use the Setnet configuration program for that. The installation script has already added an entry for demo_on in the registry so we don't need to configure that. Our objective in this Appendix is just to show the simplest way to install and use IDS with WSAD, so we do not cover this configuration through Connect.

To see the sqlhosts settings, run regedt32 in a command window to open the registry and then select HKEY_LOCAL_MACHINE -> SOFTWARE -> Informix -> SQLHOSTS:



```
HOST : REG_SZ : a23bk42f
OPTIONS : REG_SZ :
PROTOCOL : REG_SZ : olsoctcp
SERVICE : REG_SZ : demo_on_tcp
```

Figure C-14 Windows Registry

Hosts and Services files

Different than Linux and UNIX, the hosts and services files are not located in the \etc directory, they are both located in <WINNT>\system32\drivers\etc. Make sure that both files do not have any extension (such as .txt for example). IDS will only recognize these files if they do not have any extension.

TAPEDEV and LTAPEDEV configuration parameters

These are tape parameters for instance and logical logs backup. If you are not interested in taking these backups you can change these parameters to NUL, by editing your configuration file, %onconfig%. So now when the server needs a backup it will not wait for a tape device. On UNIX and Linux this value is /dev/null.

D

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

```
ftp://www.redbooks.ibm.com/redbooks/SG246948
```

Alternatively, you can go to the IBM Redbooks Web site at:

```
ibm.com/redbooks
```

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246948.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name:	*Description*
SampleApps.zip:	Zipped file containing four executable Sample Applications, a Sample Application Implementation Guide, and an implementation file.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	50MB minimum
Operating System:	Windows/2000
Memory:	756MB

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. There will be six folders. Four of the folders each contain a sample application, one contains a Sample Applications Implementation Guide, and the other contains the Master.css file required for the applications. These are the folders:

1. **SelectApplication:** Contains the sample application called “ITSOStoresDBPages” that only performs a ***select*** operation against IDS. It was developed with Database Web Pages. To install the sample demo application, follow the instructions in “Verifying the configuration” on page 87. In particular, to only install this sample demo (rather than actually developing the demo yourself) refer to the specific instructions in the Shaded Note Box, on page 88.
2. **FullApplication:** Be careful! This application has the same name as the one above in the SelectApplication folder. It is the same application base, but has been enhanced to perform additional operations. The sample application is also called “ITSOStoresDBPages” but performs a ***select, insert, update, and delete*** operation against IDS. It was also built with Database Web Pages.
3. **ITSOStores:** Contains the sample application called “ITSOStores”. It is similar to the sample application in the FullApplication folder, but is developed with different technology and has expanded capabilities. For example, it includes the use of such things as Java beans and container managed persistence. But it was developed to run in a single server environment.
4. **ITSOStoresJMS:** Contains two sample applications called “ITSOStoresJMS” and “TestJMS”. They are similar to the sample application in the ITSOStores folder, but have been enhanced to run in a distributed environment by the addition of Java Messaging Services. However, they will also run in a single server environment because distribution can be physical or logical.
5. **ImplementationGuide:** Contains a Sample Applications Implementation Guide document (.pdf) file. This document has step-by-step instructions for implementing the four sample applications listed above. Much of the same information is included throughout the redbook, but this document condenses it all into one place to make implementation faster and easier. It also includes instructions to help you actually develop the sample applications yourself, as a learning experience.
6. **MasterImp:** Contains the *Master.css* file that is used with the sample applications. You will be prompted for the file in the step-by-step implementation process.

Glossary

Attribute. In XML, a name=”value” pair that can be placed in the start tag of an element. The value must be quoted with single or double quotes.

Bandwidth. Measure of the information capacity of a transmission channel.

Bean. A definition or instance of a JavaBeans component. See JavaBeans

CGI. The Common Gateway Interface (CGI) is a means of allowing a Web server to execute a program that you provide rather than to retrieve a file. A number of popular Web servers support the CGI. For some applications, for example, displaying information from a database, you must do more than simply retrieve an HTML document from a disk and send it to the Web browser. For such applications, the Web server has to call a program to generate the HTML to be displayed. The CGI is not the only such interface, however.

Class Method. Methods that apply to the class as a whole rather than its instances (also called a static method)

Class. An encapsulated collection of data and methods to operate on the data. A class may be instantiated to produce an object that is an instance of the class.

Client. A software program used to contact and obtain data from a server software program on another computer -- often across a great distance. Each client program is designed to work specifically with one or more kinds of server programs and each server requires a specific kind of client program.

Client/Server. The relationship between servers in a communications network. The client is the requesting processor, the server the supplying processor. Also used to describe the information management relationship between software components in a processing system.

Cluster. A type of parallel or distributed system that consists of a collection of interconnected whole computers and is used as a single, unified computing resource.

Enterprise Java Bean. The Enterprise JavaBeans specification defines a way of building transactionally aware business objects in Java

Enterprise Network. A geographically dispersed network under the auspices of one organization.

Entity. In XML, an entity declaration provides the ability to have constants or replacement strings, which are expanded by a pre-processor. An entity declaration maps some token to a replacement string. Later the token can be prefixed with the & character and the replacement string is put in its place.

Factory. A bean that dynamically creates instances of beans.

Garbage Collection. Java’s ability to clean up inaccessible unused memory areas (garbage) dynamically. Garbage collection slows performance, but keeps the machine from running out of memory.

Gb/s. Gigabits per second. Also sometimes referred to as Gbps.

GB/s. Gigabytes per second. Also sometimes referred to as GBps.

Gigabit. One billion bits, or one thousand megabits.

IP. Internet Protocol.

Java Applet. A small Java program designed to run within a Web browser. It is downloadable and executable by a browser or network computer.

Java Naming and Directory Interface (JNDI). A set of APIs that assist with the interfacing to multiple naming and directory services. (Definition copyright 1996-1999 Sun Microsystems, Inc. All Rights Reserved. Used by permission.)

Java Native Interface (JNI). A native programming interface that allows Java code running inside a Java Virtual Machine to interoperate with applications and libraries written in other programming languages.

Java Server Page (JSP). Java Server Pages are Web pages that include dynamic tags which are executed on the server. JSPs are the presentation layer for Web-based applications built in Java.

Java Virtual Machine. A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

Java. An object-oriented programming language for portable, interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated. The Java environment consists of the JavaOS, the Virtual Machines for various platforms, the object-oriented Java programming language, and several class libraries.

JavaBeans. Java's component architecture, developed by Sun, IBM, and others. The components, called Java Beans, can be parts of Java programs, or they can exist as self-contained applications. Java Beans can be assembled to create complex applications, and they can run within other component architectures (such as ActiveX and OpenDoc).

JDBC (Java Database Connectivity). In the JDK, the specification that defines an API that enables programs to access databases that comply with this standard.

Local Area Network or LAN. A network covering a relatively small geographic area (usually not larger than a floor or small building).

Mb/s. Megabits per second. Also sometimes referred to as Mbps.

MB/s. Megabytes per second. Also sometimes referred to as MBps.

Method. A fragment of Java code within a class that can be invoked and passed a set of parameters to perform a specific task

Network. An aggregation of interconnected notes, workstations, file servers, and/or peripherals, with its own protocol that supports interaction.

Object. The principle building block of object-oriented programs. Objects are software programming modules. Each object is a programming unit consisting of related data and methods.

Package. A program element that contains classes and interfaces.

Persistence. In object models, a condition that allows instances of classes to be stored externally, for example in a relational database.

Protocol. A data transmission convention encompassing timing, control, formatting and data representation.

Uniform Resource Locator (URL). The unique address that tells a browser how to find a specific Web page or file.

Abbreviations and acronyms

AAT	Application Assembly Tool
AAT	Application Assembly Tool
AE	Advanced Edition
API	Application Programming Interface
AS	Application Server
AXIS	Apache eXtensible Interaction System
BLOB	Binary Large OBject
BMP	Bean-Managed Persistence
BMT	Bean-Managed Transactions
CMP	Container-Managed Persistence
CMR	Container-Managed Relationships
CMT	Container-Managed Transactions
DB2 UDB	IBM DB2 Universal Database
DBMS	Database Management System
EAR	Enterprise ARchive
EJB	Enterprise Java Beans
EJS	Enterprise Java Server
FTP	File Transfer Protocol
Gb	Giga Bits
GB	Giga Bytes
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBM	International Business Machines Corporation
IDE	Integrated Development Environment
IDL	Interface Definition Language
IDS	Informix Dynamic Server
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
ITSO	International Technical Support Organization
J2C	J2EE Connector
J2EE	Java 2 Platform Enterprise Edition
JAR	Java ARchive
JDBC	Java Database Connectivity
JDK	Java Developers Kit
JMS	Java Messaging Service
JNDI	Java Naming and Directory Interface
JRE	Java Runtime Environment
JSP	Java Server Pages
JTA	Java Transaction API
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
Mb	Mega Bits
MB	Mega Bytes
RDBMS	Relational Database Management System
SDK	Software Developers Kit
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP	Transmission Control Program
URL	Uniform Resource Locator
WAR	Web Application aRrchive
WAS	WebSphere Application Server
WSAD	WebSphere Studio Application Developer
WSDL	WebSphere Description Language
WSGW	Web Services GateWay
WSIF	Web Services Invocation Framework
WSP	WebSphere Portal
WWW	World Wide Web
XML	eXtensible Markup Language

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 350. Note that some of the documents referenced here may be available in softcopy only.

- *Web Services Wizardry with WebSphere Studio Application Developer,* SG24-6292
- *IBM WebSphere Application Server V5.0 System Management and Configuration: WebSphere Handbook Series*, SG24-6195
- *IBM WebSphere V5.0 for Linux, Implementation and Deployment Guide - WebSphere Handbook Series*, REDP-3601
- *WebSphere Application Server V4 for Linux, Implementation and Deployment Guide, REDP-0405*
- *WebSphere Studio Application Developer Version 5 Programming Guide, SG24-6957*
- *WebSphere Studio Application Developer Programming Guide,* SG24-6585
- *Web Services Version 5 Web Services Handbook,* SG24-6891
- *Linux Application Development Using WebSphere Studio 5,* SG24-6431
- *IBM WebSphere V5.0 Security WebSphere Handbook Series, SG24-6573*
- *Self-Study Guide: WebSphere Studio Application Developer and Web Services, SG24-6407*
- *WebSphere Version 4 Application Development Handbook, SG24-6134*
- *EJB 2.0 Development with Web Sphere Studio Application Developer,* SG24-6819
- *IBM WebSphere Portal V4.1 Handbook Volume 1*, SG24-6883
- *IBM WebSphere Portal V4 Developer’s Handbook*, SG24-6897

Other publications

These publications are also relevant as further information sources:

- Brown, Paul G. “An Object-Relational Approach to Building a High-Performance XML Repository”, in XML Data Management: Native XML and XML-Enabled Database Systems by Akmal B. Chaudhri, Awais Rashid, Roberto Zicari

Online resources

These Web sites and URLs are also relevant as further information sources:

- IBM WebSphere Application Server

  `http://www.ibm.com/software/webservers/appserv`
- IBM WebSphere InfoCenter

  `http://www.ibm.com/software/webservers/appserv/infocenter.html/`
- IBM Informix Product Family

  `http://www.ibm.com/software/data/informix/`
- IBM Informix Dynamic Server

  `http://www.ibm.com/software/data/informix/ids/`
- SUN Java 2 Platform

  `http://www.ibm.com/software/webservers/appserv/`
- SUN's Java 2 Platform - Enterprise Edition

  `http://www.java.sun.com/j2ee`

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

Index

A

Action events 269
Administration Repository 118
Administrative Console 112, 115
Administrator-controlled redirection 10
Aggregation Module 259
Apache AXIS framework 234
    Data Encoding support 235
    Flexible Messaging Framework 234
    Flexible Transport Framework 234
Apache Jetspeed 258
Apache web server 35
Application client container 73
Architecture - ITSO Stores sample application 87
Asynchronous Input/Output 5
Asynchronous replication 11
Authentication Server 259
Automatic Redirection with the DBPATH 10
AXIS installation and preparation 236

B

bean-managed transactions 168
Binary Large OBjects (BLOB) 6, 184
    Blobspaces 6
Binding 211
BLOB 6, 184
B-tree 23
    cleaner queue 24
    cleaner thread 24
    scanner 24
buffer pool 5
Built-in data types 16
business-to-business (B2B) 254, 257
business-to-consumer (B2C) 254, 257
business-to-employee (B2E) 254

C

cells directory 119
Central Processing Unit 5
Changing Business Environment xix
Character Large OBjects (CLOB) 6
checkpoint duration 19
Chunks 5
class libraries 194
Class location 92
Client-Server Connectivity 6
CLOB 6
Collaboration services 260
collaborative components 256
Collection Data Types 16
Communications Gateway Interface (CGI) 65
Complex Data Types 15
configuration data 126
configuration directory 119
Connecting IDS and WebSphere 130
    Data Source object 131
Connection factory JNDI name 161
Connection name 92
Connection Page 91
Connectionless Computing 65
Container Managed Persistent Beans 88
Container-managed transactions 168
Content Management 260
cookies 65
customer entity bean 152
customer relationship management (CRM) 254

D

DADX - document access definition extension 221
    build a DADX Web service 222
    Create a DADX group 224
    DADX functions 221
    File Format 327
    generated DADX file 226
    runtime component 221
    support for stored procedures 229
    support for user defined routines 229
    Web service deployment 233
data manipulation language (DML) 18
data replication 11, 127
Database connections 66
Database Server 4
    Client-Server Connectivity 6
    configuration file 7
    Process Component 5
    Shared Memory Component 5
Database Web Pages 88
    Steps to develop a sample application 88
DataBlade 13, 182, 184, 234
    DataBlade API 279
DBSERVERALIASES 6
DBSERVERNAME 6
Dbspaces 6
demo html/page 200
Deploying IDS and WebSphere 122
Destination JNDI name 161–162
Dirty count 25
disk mirroring 127
Document Type Definition 176
DOM (Document Object Model) 183
driver manager connection 95
DTD editor 186
Dynamic Scalable Architecture (DSA) 3

E

Eclipse 77
    Eclipse platform 77

Eclipse Workbench technology 78
Eclipse/WebSphere Studio workbench 77
integrated development environment (IDE) 78
perspectives 78
primary perspectives 79
Role-based development model 78
EGL 292
automatically translate I4GL code 292
moving I4GL code to EGL 294
Non-procedural Statements 293
procedural statements 292
Reports 294
translating I4GL form files 293
Type System 292
EJB transaction demarcation 168
Enable administration client 160
Enterprise application project name 89
Enterprise Generation Language (EGL) 84, 291
Enterprise Replication (ER) 11
enterprise resource planning (ERP) 254
Entity ARchive file (EAR)
EAR Export 102
Excalibur Text Data Blade 184
Executive Summary xiii
Expression Builder 93
extended data types 16
Extents 5

F
Firewall on SuSE 299
adding new ports 300
Flyweight pattern 263
Four J’s 84
Fragmentation 8
Fuzzy Checkpoints 19

G
Global transaction 167
Guidelines for Using Transactions 172

H
HDR 127
High-Availability Data Replication (HDR) 9
Archive and Restore 9
Balanced I/O 9
connection pooling 129
Considerations 125
failures, and actions taken 127
HDR now supports replication 26
Higher Availability 9
IDS high availability and WebSphere 127
JDBC support 128
Parallel scans 9
High-Performance Loader (HPL) 13
host name 109
Host variable name 94
Hot list 25
HTML page 197
HTTP port number 97
HTTP server 73

I
I4GL 291
IBM Content Manager 257
IBM DB2 UDB 255
IBM Lotus 256
Collaborative Components 256
Collaborative Places 256
Extended Search 256
QuickPlace 257
Sametime 257
IBM SecureWay Directory 255
IBM Tivoli 12
Site Analyzer 256
Storage Manager (TSM) 12
Tivoli Access Manager 257
IBM Web Content Publisher 255
IBM WebSphere Application Server 255, 258
IBM WebSphere Personalization 255
IBM WebSphere Portal 254
and IDS 276
architecture 258
concepts 261
Enable 254
Enable components 255
engine 259
Experience 254
Experience components 257
Extend 254
Extend components 256
for Multi platforms 254
Industry research 257
infrastructure 260
Jetspeed implementation 259
LDAP 260
Model-View-Controller (MVC) design pattern 266
Page aggregation 270
Portlet events and messaging 268
portlet lifecycle 267
Portlet solution patterns 273
portlets 262
services 260
Toolkit 261
Implementation Hints and Tips 295
Implementation Scenarios 123
Single Server Model 123
Three-Server Model 125
Two-Server Model 124
Informix 4GL 84, 289, 291
Protecting your Investment 290
Informix and WebSphere - Why? xx
Informix C-ISAM 2
Informix Dynamic Server xiv, 1, 121
9.40 features 22
ANSI join syntax 28
Architecture 3
The Big Picture 8
as a Web service consumer 233

Chunks redirection 30
Command line chunk mapping 30
Configuring IDS 21
Creating the users and groups 32
Data availability enhancements 26
Enhanced SELECT statement syntax 28
environment variables 32
Extended Features of IDS 9.x 13
features 2, 8
Full use of storage media 30
functionality on Windows 340
I-Connect and JDBC 40
Increased file size limit 30
Increased maximum size of chunks 22
Increased number of dbserver aliases 23
Informix Server Administrator (ISA) 43
install and configure on Linux 31
Installation directory 32
installation verification 41
Installing and Configuring 21
installing on Linux 33
Linux hardware and software recommendations 31
Linux Kernel parameters 31
Linux port number 92
LRU settings 26
Memory management and tuning 25
Optimized index cleaning 23
Performance enhancements 23
priority management for the buffer manager 25
product directory 34
scalability and usability enhancements 22
Security enhancements 22
Sequence Objects 29
Stored procedures return parameters 28
stores_demo database 42
TCP/IP connection protocol 42
Triggers on views 27
Windows/Linux Differences 340
with J/Foundation 2, 235
Informix Extended Parallel Server (XPS) 3
Informix JDBC driver 87
Informix JDBC Provider 114
Informix MQSeries DataBlade 286
Informix Object Translator 185
Informix Online Extended Edition 2
Informix Rapid Application Development (IRAD) 84
Informix SE 2
Informix Spatial DataBlade 185
Informix Storage Manager (ISM) 12
Informix's Virtual Table Interface 286
INFORMIXSERVER 6
Input Form 101
Insert on a view 27
Installing WebSphere Application Developer 85
Installing WSAD on Windows/2000 86
Integrated Development Environment (IDE) 84
Internal JMS Server 162
IP ports 104
ipload utility 13
IPX/SPX 6
Isolation levels in JDBC 170
iterator function 246
itso - the generic user 37
ITSO stores sample application architecture 87
ITSOStoresDBPages 91, 116–117
ITSOStoresDBPagesWeb 91, 97

J
J2EE 68, 258
application and programming model 69
Components 71
Connectors 71
Containers 71
First Tier (View) 70
Middle Tier (Control) 70
Third Tier (Model) 71
Benefits 69
Hierarchy 89
Overview 68
Perspective 89, 224
Java 66
Enterprise Edition 68
Enterprise Java Beans (EJB) 67
Informix JDBC driver 87
Informix JDBC Provider 114
Java Bean XML/XSL client wizard 204
Java beans from a DTD 205
Java beans from an XML schema 206
Java classpath 238
Java Database Connectivity (JDBC) 66
Java Message Services (JMS) 88, 283
Asynchrounous Messaging 283
Embedded JMS 284
embedded JMS server user 104
External JMS/MQ 284
Lightweight JMS 283
Java package 91
Java Server Pages (JSP) 67, 88
Java servlet creation 194
Java servlets 72
Java Transaction API 168
Java UDR deployment 308
Java Virtual Machine (JVM) 66, 235
Java Virtual Processors 235
Java wrapper UDR 247
Java-enabled ISA features 38
JNDI name 154
servlet engine install 38
Java Virtual Machine (JVM) 235
JAXP 183
JDBC Provider 114
JMS server user 104
JMS Servers 162

L
LDAP 254
Legacy data 126
lightweight directory access protocol (LDAP) 254
Lightweight JMS 283

Linux 305
Determining the port number of IDS 305
launching the installation script 307
WebSphere Application Server 307
Listener Ports 161
local transaction 167
log data capture 11
Logical Input/Output 5
Logical Log 6

M
memory usage 18
Message events 269
Message Listener Service 161
Message Queue Interface 280
Message-driven beans 171
Model 91
MQ Event Broker 285
MQBACK 286
MQCMIT 286
multiple devices 258
Multithreading 3
Implementation 3
multi threaded process 3
MVC Architecture 67

N
NAMED Row Type 16
create a hierarchy 17
Creating and Manipulating 17
node name 109

O
Object-Relational DBMS (ORDBMS) 13
ON-Bar 11
components 12
onconfig 7
onpload database 13
onpload utility 13

P
Page aggregation 270
page group 262
Pages 5
Parallel Backup and Restore 11
Performance tuning guidelines 303
DSS tuning 304
OLTP tuning 304
Permissions 262
Persistent storage 126
Personalization 260
Physical Input/Output 5
Physical Log 6
Place 262
Portal 253
page structure 259
portal administrator 262
Portal engine 259
portal page 262
Portal Server 67
Portal Servlet 259
Portlet 261
Portlet application 261
Primary-Target 11
Process Component 5

Q
Query Builder 190
Queue Manager 281
Clustering 285

R
read-only users 37
Redbooks Web site 350
Contact us xviii
Redhat 8.0 Linux 308
Install Error 308
Row Data Types 16

S
sample application xiv, 88, 144
Architecture 87
FullApplication 144–145
Creating a JSP 149
Database Web Pages 145
Deploying in Application Server 152
Deploying in WSAD 151
search, insert, update, and delete 145
steps to extend 145
Steps to integrate 149
ITSOStores 88, 152
deploy in WAS 155
Deploying in WSAD 154
Generating Deploy and RMIC Code 153
Java Server Pages (JSP) 152
Servlets 152
Steps for creating a CMP bean 152
using container managed persistent bean 152
ITSOStoresJMS 158
Container Managed Persistent Bean 160
Data source 162
Deploying in WAS 162
Deployment in WSAD 160
design 158
distributed environment 158
Java Message Services 158
Java Server Page 160
JNDI names 161
Listener Port 161
Message Driven Bean 160
Message Oriented Middleware 159
preparation work 160
queue 161
Queue Connection factory 161
two sample applications 159
WebSphere MQ 158

names 144
FullApplication 144
ITSOStores 144
ITSOStoresJMS 144
SelectApplication 144
SelectApplication 88
Database Web Pages 88
Deploying in test server 99
Deploying the application 102
Running on application server 100
Steps to develop 88
Save to Master Configuration 116
SAX (Simple API for XML) 183
Sbspaces 6
Scalability 122
Search 260
sequence objects 307
serial data type 307
Server 97
Components 162
Configuration 97
connection 6
Instance 97
perspective 160
Service roles and interactions 211
Service brokers 211
Service clients 211
Service providers 211
setting the installation directories 109
Shared Memory 6
Component 5
Simple Object Access Protocol (SOAP) 83, 211
Site Analysis 260
Skins 262
Skins page 262
Smart large objects 6
Socket and Transport Layer Interface (TLI) 6
SQL 18
Statement Cache 18
Statement Type 91
syntax for SEQUENCE objects 29
sqlhosts file 7
SQLResult 193
SQLtoXML 187
sample project 188
template file (.xst) 191
Starting the test server 100
Steps to install WSAD from CD-ROM 86
Steps to install WSAD from electronic image 86
Storage Manager 12
Stream Pipes 6
supply chain management (SCM) 254
Systems Architecture
illustration 130
used in our installation 129

T
Table and Index Partitioning 8
table hierarchy - creating one 17
Tablespaces 6
tar file 33, 105
TCP/IP 6
Monitor 220
Themes 262
Threads - dynamic configuration 24
Toolkit 261
transaction attribute 171
Transactions in EJB 166
Transcoding Technology 260

U
UDDI 83, 211
UDR 14
UDR - Creating it 15
UDT 6, 15
Update-Anywhere 11
User-controlled redirection 10
User-Defined Data Types (UDT) 6, 15–16
User-Defined Routines (UDR) 14

V
Verity Text Search Data Blade 184
Virtual Network Computing 297
VNC server 297
VNC viewer 297
Virtual Processors 5
VisualAge Generator 291

W
Web DataBlade 183
Web server plug-in 113
Web Services 209, 261
a complex example - Google Search 245
based on EJBs 214
based on Java beans 215
best practices 213
Categories 210
consumer requirements for IDS 234
Create a new Java bean 215
Currency Exchange example 239
Generate the WSDL files 217
IDS as a consumer 233
IDS as a provider 214
introduction 210
monitoring 219
Service roles and interactions 211
standards 211
tools for development 213
use of IBM Informix Dynamic Server 209
Web Services Description Language (WSDL) 83, 211
Web Services Inspection Language (WSIL) 83, 211
Web Services Object Runtime Framework (WORF) 221
WebSphere xiv, 63–64, 66
administration repository 118
Collaboration services 260
configuration data 126
configuration directory 119

JMS Provider 161
Product Family 67
Business Integration 67
Foundation and tools 67
Reach and User Experience 67
Queue Destinations 161
WebSphere Application Server xiv, 67–68, 103, 255
Architecture 71
benefits 74
Configuring for use with IDS 114
Deploying the application 102
EJB container 72
Enterprise Application Server 64
Hardware requirements 76
Installation from CD-ROM 106
Installation from tar file 105
Installing 105
Installing on SuSE Linux V8.0 104
Operating System and Processor requirements 76
Packages 73
Enterprise Edition 74
Express 73
Network Deployment 74
Pre-installation tasks 104
Product Package Comparisons 74
Verify Installation 111
Web container 72
WebSphere JMS Provider 161
WebSphere MQ 67, 279
Event Broker 285
features at a glance 280
Integration with IDS 285
JMS Provider 285
Message Queue Interface (MQI) 280
Putting and getting messages 282
queue manager 281
Transactional support for IDS 286
WebSphere Portal 254–255, 259–260
architecture 258
Client to remote application 274
collaborative components 256
configure IDS 277
infrastructure 260
Java portal engine 259
Packages 254
page 262
page aggregation processing 272
page layout 262
Portal concepts 261
portlet 261
Portlet application 261
Portlet container 259
presentation services 259
Server 253
IDS as a repository database 253
services 260
Skins 272
templates 272
Themes 272
tooling 261
WebSphere Portal Enable 254–256
WebSphere Portal Experience
advanced collaboration 257
content management functions 257
features 256
WebSphere Portal Extend 256
WebSphere Portal Packages
Portal Enable 254
Portal Experience 256
Portal Extend 256
extended search capabilities 256
Web analysis technology 256
WebSphere Studio Application Developer xiv, 67, 77
Configuring for IDS 85, 87
Eclipse 77
highlights 82
Installing and Configuring 85
Installing from the CD-ROM 86
Installing from the electronic image 86
integrate existing tools 77
integration with Informix 4GL 84
Packages 80
Application Developer 80
Enterprise Developer 80
Integration Edition 80
Site Developer 80
Pre-installation tasks 86
Test with IDS 87
Verifying the Configuration 87
WebSphere test environment 97
Configuring 98
Deploying the web application 99
Running the sample application 100
Window events 269
Windows 333
configuration of IDS 333
Hosts and Services files 342
JDK and JDBC installation 339
SQLHOSTS 341
Workflow 11
World Wide Web 65

X

XML 175
Binary Large OBjects 184
DataBlade 182
document field 203
Document Management 177
Document Types 178
dynamic XML 181
editor 186
editor tools 186
format conversion 199
generate DDL 208
generate schema from table 207
Generation and Transformation with Stored Procedures 182
Getting XML In 180
Getting XML Out 181
Java API 183

Java Bean XML/XSL client wizard 204
key features 176
mapping options in WSAD 204
Queries 180
Schema 176
schema editor 186
SQL query wizard 187
standards 178
Storing XML 181
Support in Database Systems 180
typical applications 177
Usage Scenarios 177
Web DataBlade 183
What’s available with IDS 181
XSL Debug 187
XSL documents 182
XSL stylesheet 195
XSL transformation 187
XMLtoSQL 187
XPath expression wizard 186

Using Informix Dynamic Server with WebSphere

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Using Informix Dynamic Server with WebSphere

Informix Dynamic Server for data management

WebSphere for application development productivity

Solution integration and easy implementation

This IBM Redbook is intended primarily for Informix customers and business partners. It presents information that will help with the installation, configuration, use, and management of an Informix Dynamic Server and WebSphere environment.

We provide a brief overview of Informix Dynamic Server, WebSphere Application Server, and WebSphere Studio Application Developer, all in one place, for a better understanding of the products. We then show how these products can be integrated together, thus enabling you to begin receiving the benefits of a powerful e-business application development environment as quickly as possible.

Informix Dynamic Server is supported by WebSphere, an open application development platform that can reduce e-business application development timeframes by providing a common toolset that can enhance developer productivity.

SG24-6948-00 ISBN 0738453498