

WebSphere Commerce V5.4 Developer's Handbook

Understanding the development
process

Planning and using a
development environment

Customization examples



Bill Moore
Peter Gothager
Michael Mattinson
Chiara Montecchio
Narayan Prasad
Carla Sofia Jesus Ribeiro
Thomas Tolborg



International Technical Support Organization

WebSphere Commerce V5.4 Developer's Handbook

July 2002

Take Note! Before using this information and the product it supports, be sure to read the general information in “Notices” on page ix.

First Edition (July 2002)

This edition applies to version 5.4 of WebSphere Commerce Business Edition and WebSphere Commerce Professional Edition, for use with the Windows NT and 2000 operating systems.

This document created or updated on October 18, 2002.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2002. All rights reserved.. All rights reserved.
Note to U.S Government Users – Documentation related to restricted rights – Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this redbook	xi
Comments welcome	xiii
Part 1. Store development basics	1
Chapter 1. Introduction	3
1.1 Overview	4
1.2 Who should read this book	6
1.3 Organization	7
Chapter 2. Architecture and programming model	11
2.1 Store architecture overview	12
2.1.1 Store assets	12
2.1.2 WebSphere Commerce store architecture	13
2.2 WebSphere Commerce application architecture	14
2.2.1 WebSphere Commerce server subsystems	18
2.2.2 Data assets	19
2.3 WebSphere Commerce runtime architecture	24
2.3.1 WebSphere Commerce runtime components	24
2.3.2 Topology configuration samples	31
2.4 WebSphere Commerce programming model	39
2.4.1 The J2EE Programming Platform	40
2.4.2 Model-view-controller design pattern	40
2.4.3 Persistent Object Model Overview	45
Chapter 3. Requirements and design	49
3.1 Application development methodology	50
3.1.1 Solution startup	51
3.1.2 Solution outline	51
3.1.3 Macro design	53
3.1.4 Micro design	54
3.1.5 Build cycle	55
3.1.6 Deployment	57
3.1.7 Solution Close	57
3.2 Example store requirements	57

3.2.1	General store requirements	58
3.2.2	Quick order implementation	61
3.2.3	Price display including tax	62
3.2.4	Product creation via MQ	63
3.2.5	Welcome page based on role	64
3.2.6	Amount-based order approval	65
3.2.7	Contract-based logon	67
3.2.8	Product bundles	68
Chapter 4. Planning and development		71
4.1	Development overview	72
4.1.1	WebSphere Commerce Studio V5.4	72
4.2	Planning WebSphere Commerce development	76
4.2.1	Development initialization phase	76
4.2.2	Team development	77
4.3	Install the development environment	79
4.3.1	Pre-installation requirements	79
4.3.2	Install VisualAge for Java, Enterprise Edition	81
4.3.3	Install WebSphere Studio	84
4.3.4	Install WebSphere Commerce Studio	85
4.3.5	Install Application Assembly Tool	88
4.4	Post-install configuration	90
4.4.1	Configure VisualAge for Java, Enterprise Edition	90
4.4.2	Create a sample store in VisualAge for Java	95
4.4.3	Maintenance	95
4.5	WebSphere Studio Application Developer	96
4.5.1	Introduction	97
4.5.2	Setup WebSphere Studio Application Developer	101
4.5.3	Setup WebSphere Test Environment	113
4.5.4	The second installation	120
4.5.5	Developing in WebSphere Studio Application Developer	123
Chapter 5. Creating a store		131
5.1	Stores architecture overview	132
5.1.1	Store assets and components	132
5.1.2	Store design	133
5.2	Store development	134
5.2.1	Creating a store based on sample stores	134
5.2.2	Creating store by generating new assets	166
5.2.3	Create store using a mixed approach	168
Chapter 6. Testing a store		169
6.1	Testing strategy	170
6.2	Test planning	171

6.2.1	The test administrative plan	171
6.2.2	Problem management	171
6.2.3	Version control	172
6.3	Test phases	173
6.3.1	Static test	173
6.3.2	Unit testing	179
6.3.3	Functional test	181
6.3.4	System test	182
6.4	Test case design	185
6.4.1	Test cases samples	185
6.5	Test environment set-up	191
6.5.1	Staging server and test data	192
6.6	Problem determination	194
6.6.1	Tracing	195
6.6.2	Error handling	197
Chapter 7. Packaging and deployment		201
7.1	Overview	202
7.2	Using WebSphere Studio and Store Services	202
7.3	Using Ant to package and deploy a store	203
7.3.1	Folder structure	203
7.3.2	Using the sample Store Archives	204
7.3.3	The Deployment Use Case	205
7.3.4	Creating the Ant build files	207
7.3.5	Conclusion	213
Part 1. Customization examples		215
Chapter 8. Examples overview		217
8.1	Example stores	218
8.2	Orders	218
8.2.1	CICS order transaction	218
8.2.2	Quick order function	219
8.3	Shipping and taxes	221
8.3.1	Shipping by weight	221
8.3.2	Display prices with tax	222
8.3.3	Discounts	223
8.4	Messaging customization	224
8.4.1	Inbound MQSeries - product creation	224
8.5	B2B features	226
8.5.1	Role-based display	226
8.5.2	Amount-based order approval	228
8.5.3	Contract-based shopping	228
8.6	Product entry and display	230

8.6.1	Product comparison	230
8.6.2	Enabling bundles	231
8.6.3	Display of multiple currencies	231
Chapter 9.	Orders	233
9.1	CICS order transaction	234
9.1.1	Functional requirements	234
9.1.2	Use cases	235
9.1.3	Design	236
9.1.4	Pre-requisites	238
9.1.5	Create the Task Command	243
9.1.6	Register the command in WebSphere Commerce	255
9.1.7	Deploy the code	255
9.2	Quick Order	256
9.2.1	Quick order flow in the ToolTech store	256
9.2.2	Design	258
Chapter 10.	Shipping and taxes	259
10.1	Shipping by weight	260
10.1.1	Example of shipping calculations by weight	260
10.1.2	Shipping database assets	261
10.1.3	Adding shipping by weight charges	263
10.1.4	Use case for order check out	265
10.2	Prices including taxes	270
10.2.1	Definition of tax types	270
10.2.2	Tax database assets	271
10.2.3	Implementation prices including taxes	274
10.3	Discounts	282
10.3.1	Discount types	283
10.3.2	Discount assets	283
10.3.3	Creating a discount in a sample store	285
Chapter 11.	Messaging customization	291
11.1	Installing and configuring MQSeries	292
11.1.1	Installing	292
11.1.2	Configuring	292
11.2	Enabling the MQ adapter in WebSphere Commerce	296
11.2.1	TransportAdapter	296
11.2.2	Log level	297
11.3	Enabling the MQ adapter in VisualAge for Java	297
11.3.1	TransportAdapter	297
11.4	Functional requirements	299
11.4.1	Use case	299
11.5	Preparing for the MQProductCreate command	300

11.5.1	Defining XML and DTD for the command	300
11.5.2	Registering the DTD in WebSphere Commerce	305
11.5.3	Creating mapping between the XML and the command	305
11.5.4	Registering the command in WebSphere Commerce	308
11.6	Creating the commands	309
11.6.1	Creating the MQProductCreateCmd interface	309
11.6.2	Creating the MQProductCreate command impl class	309
11.6.3	Creating the MQProductAttributesCmd interface	316
11.6.4	Creating the MQProductAttributesCmdImpl class	316
11.7	Testing the MQProductCreate command	320
11.7.1	Deploying the commands	321
11.7.2	Modifying JSP files	321
11.8	Final considerations	323
11.8.1	The markfordelete attribute	323
11.8.2	Inventory	323
11.8.3	Performance	323
11.8.4	SKUs and attributes	324
11.8.5	Transactions	324
Chapter 12. B2B features		325
12.1	B2B features in WebSphere Commerce Business Edition V5.4	326
12.1.1	Access control	326
12.2	Role-based display	327
12.3	Order approval	331
12.4	Contracts and trading agreements	338
12.5	Message extensions	345
12.5.1	cXML overview	346
12.5.2	cXML in WebSphere Commerce Business Edition	347
Chapter 13. Product entry and display		349
13.1	Product comparison	350
13.1.1	Product Advisor	350
13.1.2	Creating base search space	350
13.1.3	Preparing a product comparison metaphor	361
13.1.4	Testing product comparison	366
13.2	Products and bundles	367
13.2.1	Definition of products	367
13.2.2	Creating new product with product manager	368
13.2.3	Testing the new product	375
13.2.4	Definition of bundles	377
13.2.5	Bundles in WebFashion	377
13.2.6	Creating bundles	378
13.3	Display of multiply currencies	386

13.3.1 Currencies types	387
13.3.2 Dual display and counter values	389
13.3.3 Implementation of dual display of currencies	390
13.3.4 Testing the dual display of currencies.	394
Chapter 14. Migration examples	397
14.1 Migration Overview	398
14.2 Migration considerations	398
14.2.1 Infrastructure changes	399
14.2.2 Architecture changes	400
14.2.3 Database changes	402
14.2.4 Additional considerations	404
14.3 Migration example(s?)	406
Appendix A. Sample level 1 “a.” appendix heading (yHead0Appendix)	407
Sample level 2 heading (yHead1Appendix), new page	408
Sample level 2 heading (yHead2Appendix)	408
Sample level 3 heading (yHead3Appendix)	408
Appendix B. Additional material	409
Locating the Web material	409
Using the Web material	409
System requirements for downloading the Web material	410
How to use the Web material	410
Abbreviations and acronyms	411
Related publications	415
IBM Redbooks	415
Other resources	415
Referenced Web sites	416
How to get IBM Redbooks	417
IBM Redbooks collections.	417
Index	419

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:

CICS®

DB2®

DB2 Universal Database™

Encina®

IBM®

iSeries™

Metaphor®

MQSeries®

MVS™

OS/390®

Redbooks™

S/390®

SP™

TeamConnection®

Tivoli®

VisualAge®

WebSphere®

zSeries™

Redbooks(logo)™ 

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

Lotus®

eSuite™

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare 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.

C-bus is a trademark of Corollary, 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 redbook details the new tools and techniques available to J2EE developers using WebSphere Commerce V5.4 to customize e-commerce shopping sites.

It provides worked examples of customizations proceeding from realistic requirements through to modification of store assets, including front end, data and back end assets. Customization examples include both simple customization and complex programming customizations.

The team that wrote this redbook

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

Bill Moore is a WebSphere Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes on WebSphere and related topics. Before joining the ITSO, Bill was a Senior AIM Consultant at the IBM Transarc lab in Sydney, Australia. He has 17 years of application development experience on a wide range of computing platforms and using many different coding languages. He holds a Master of Arts degree in English from the University of Waikato, in Hamilton, New Zealand. His current areas of expertise include application development tools, object-oriented programming and design, and e-business application development.

Peter Gothager is an IT Specialist in Business Innovation Services of IBM Sweden. He has 4 years of experience in the e-business design and development in both B2B and B2C facing projects. He holds a B.Sc. degree from Chalmers University of Technology, Sweden. His current areas of expertise include application e-business design and development using the IBM WebSphere family of products.

Michael Mattinson is an IT Architect for IBM Global Services in the United States. He has five years of experience in the architecture, deployment and maintenance of e-business sites for various companies and four years of experience with IBM's Web commerce applications. In his six years with IBM, he has participated in many large IBM e-commerce initiatives. He holds a degree in Computer Science and Mathematics from New York University. His areas of expertise include e-business application architecture, e-commerce solutions and Web infrastructure technologies.

Chiara Montecchio is a Software Engineer at the IBM Tivoli Lab in Roma, Italy. She has 4 years of experience in product and application design, developing and testing. She holds a degree in Physics from the University “La Sapienza” of Rome. Her areas of expertise include Web technologies, Java programming, and implementation of e-commerce solutions with WebSphere Application Server and WebSphere Commerce. She has written extensively on architecture and programming model and on testing a store.

Narayan Prasad is a software engineer with IBM India. He has four years of experience in e-business development.

Carla Sofia Jesus Ribeiro is an e-business integration consultant with IBM Global Services in Mainz, Germany. She has two years of experience in e-business field working on IBM e-business projects in Germany. Carla holds a degree in Electrical Engineering from the University of Applied Sciences of Trier, Germany. Her areas of expertise includes WebSphere Commerce and e-business.

Thomas Tolborg is an I/T Specialist with IBM Global Services, Business Innovation Services in Aarhus, Denmark. His areas of expertise includes J2EE and e-commerce. Thomas has worked with IBM e-commerce products since Net.Commerce version 3. In the past year Thomas has been working on several WebSphere Commerce V5.1 implementations.



Figure 0-1 The redbook team: Thomas Tolborg, Michael Mattinson, Peter Gothager, Carla Sofia Jesus Ribeiro, Narayan Prasad. Absent: Chiara Montecchio (pictured below)



Figure 0-2 Chiara Montecchio

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

Daniel Dunn
Bob Fraser
Tim Fors
Craig Henkle
Mark Ho
Khalyl Khan
Eric Koeck
Carol Liang
Scott Ripley
IBM Canada

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 the address on page ii.



Part 1

Store development basics

In this part we provide an overview of the process for planning, designing and developing a site using WebSphere Commerce.



Introduction

This IBM Redbook is designed for the developer who wants to extend the functionality of WebSphere Commerce Business Edition Version 5.4. It will take you from the design of a commerce site to the authoring of custom code. This chapter provides a description of the redbook's contents.

1.1 Overview

Now more than ever businesses are feeling pressure to show results from their e-business presence. With the increased scrutiny on return on investment and the bottom line, a company's Web site must project the appropriate image to customers, partners and the world at large. Nowhere is this more true than in the commerce and procurement arena.

As recently as a couple of years ago, consumers were content with having basic commerce abilities: browsing a catalog, selecting their chosen products and paying securely using their credit card. Recently, though, e-commerce has grown from buyers conducting these rudimentary transactions to a wholesale redesign of the relationships between partners, suppliers and customers. Consumers and businesses have come to have certain expectations of commerce sites; among them a personalized experience, accommodation of cultural mores and language, quick access to repeatable orders and access via pervasive devices such as mobile phones and personal digital assistants. Companies must now use technology to streamline their business models and increase efficiency leading directly to lowered costs and closer, more rewarding customer relationships.

WebSphere Commerce Business Edition Version 5.4 provides an enterprise platform to meet the next generation commerce requirement. Built on a component-based architecture, WebSphere Commerce supports open standards allowing for more rapid application development, integration and deployment and improved maintenance. It is written entirely in Java using Enterprise JavaBeans, Java Servlets, JavaServer Pages resulting in an easily extensible product. Some of the more noteworthy aspects of WebSphere Commerce Business Edition are:

- ▶ The rich data and object models that support many different e-commerce data structures and functions. The B2B model, in particular, is very well-represented in WebSphere Commerce Business Edition Version 5.4.
- ▶ A messaging transport layer that supports both inbound and outbound messages for notification and back-end integration.
- ▶ Support of multiple languages and currencies which extends a store's reach into the global market.
- ▶ Flexible and customizable access control based on a hierarchical policy and user roles.
- ▶ Personalized recommendations based on analysis of shopper activities.
- ▶ Pervasive device support.

WebSphere Commerce Business Edition has a very strong focus on business-to-business practices. The built-in B2B functionality includes: contract-based pricing and purchasing, entitlement-based browsing, multiple shipping addresses, buyer approval and Request for Quotation (RFQ) creation. We examine and extend these aspects of the product in Chapter 12, “B2B features” on page 325.

WebSphere Commerce also provides a suite of tools that can help store creators and administrators manage their stores:

- ▶ The Commerce Suite Accelerator is used to manage the operation of an online store. It provides an integration point for all functions available to the WebSphere Commerce store administrator, including product management, marketing and customer orders.
- ▶ The Administration Console sets up site and store features such as access control, performance monitoring, messaging, payment systems and personalization.
- ▶ Store Services allows store developers and administrators to create, edit and publish stores.
- ▶ The Configuration Manager is used to create and manage WebSphere Commerce instances.

The objective of this redbook is twofold: to present an outline of a WebSphere Commerce Business Edition Version 5.4 development cycle and to provide real-world examples of programming customization for commerce sites. We start by reviewing the WebSphere Commerce Business Edition programming model and proposing a best-practices approach to site planning and creation. Next, we propose a testing methodology that can be used in an actual application setting and explain store packaging in preparation for launch. Finally, we present a series of customized solutions to possible customer requirements.

The examples show how to customize stores to add and alter functionality such as order processing, taxation, shopping flow and messaging. They can serve as templates for extending your own commerce solutions as they demonstrate how to weave custom code into the powerful base functionality of WebSphere Commerce.

While we review the use of WebSphere Commerce tools and programming models, we do not cover the installation and configuration of the product or its associated development tool suite. Information on these topics can be found at the IBM WebSphere Commerce Business Edition Version 5.4 technical library Web site:

http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

The documents at this site cover the entire spectrum of WebSphere Commerce, from general introduction to specific low-level detail. Some of the useful materials that can be found here are:

- ▶ *IBM WebSphere Commerce Programmer's Guide Version 5.4*
- ▶ *IBM WebSphere Commerce for Windows NT and Windows 2000 Installation Guide for use with DB2 Universal Database Version 5.4*
- ▶ *IBM WebSphere Commerce Store Developer's Guide Version 5.4*
- ▶ *IBM WebSphere Catalog Manager User's Guide Version 5.4*
- ▶ *IBM WebSphere Commerce Database Schema*
- ▶ *What's New in WebSphere Commerce Version 5.4*

1.2 Who should read this book

There are several high-level roles defined for WebSphere Commerce activity. They are:

Store architect	A store architect must be able to take the merchant's business requirements and map them to WebSphere Commerce function.
Store developer	Store development is performed by several subgroups including: web designers, who craft the look and feel of the site; commerce designers, who extend WebSphere Commerce Business Edition functionality using custom code; and database developers, who produce optimized SQL for business logic.
Store administrator	Responsible for store functionality, the store administrator configures commerce system items relating to messaging, rules services, payment processing and overall store information. Store administration can be broken into several subgroups: marketing managers, who manage sales, discounts and auctions; order clerks, who process orders and payments; customer service representatives, who assist shoppers; and merchants, who are responsible for overall store function.
Catalog designer	An expert on the product domain, the catalog designer creates and manages the catalogs, pricing schemes and shopping metaphors.

While all of these groups can realize some benefit from this redbook, it is primarily designed for members of the first two: architects and developers. IT architects and specialists in the Web design and programming disciplines will find information about planning and creating commerce solutions using WebSphere Commerce Business Edition Version 5.4 and extending its basic functionality.

Architects will find the first part of the book useful as it is centered largely around the concepts of solution design, planning and testing. Developers, on the other hand, will likely focus on the customization examples in the second part of the book. Refer to 1.3, “Organization” on page 7 for specific areas of interest.

1.3 Organization

This redbook is divided into two principal parts based on intended use. The first part examines the components of store planning, creation, testing and packaging and explains the parts that may be involved in each of those steps. The second part of the book contains applicable programming examples that extend the functionality of WebSphere Commerce.

Part 1, “Store development basics” on page 1 contains background and overview information on basic store planning design and development. Testing and deployment information are also included.

Chapter 2, “Architecture and programming model” on page 11 provides an outline of the architecture and programming model of WebSphere Commerce Business Edition Version 5.4. It contains information about the functionality of each component included in WebSphere Commerce and how the components interoperate. The programming model employed for the examples is also described.

Chapter 3, “Requirements and design” on page 49 describes the early stages of the development process used to create the examples shown later in the book. We detail the requirements gathering and design processes that are part of the methodology that is used by IBM and many IBM business partners in the creation of e-business solutions.

Chapter 4, “Planning and development” on page 71 focuses on the development environment setup. We outline the up-front work that is commonly required for a WebSphere Commerce Business Edition Version 5.4 development effort and the associated tool suite. Short descriptions of WebSphere Commerce Studio, WebSphere Studio Application Developer and VisualAge for Java, Enterprise Edition are included.

Important: We document a way in which WebSphere Studio Application Developer V4 can be used for development with WebSphere Commerce Business Edition V5.4, but please remember that developing WebSphere Commerce Business Edition applications with WebSphere Studio Application Developer is not officially supported at this time.

In Chapter 5, “Creating a store” on page 131, we review the assets that comprise a WebSphere Commerce site and the activities involved with creating and launching one. This chapter defines a store archive (SAR) and examines the steps necessary to create a sample store. Additionally, the use of Store Services is reviewed.

Chapter 6, “Testing a store” on page 169 introduces a testing methodology that can be utilized with an enterprise commerce site. Test planning and execution are critical to the launch of a usable and reliable commerce solution. Here we review the various stages of a valid testing strategy and the different types of testing that can be performed in the course of a launch including:

- ▶ Unit testing
- ▶ Functional verification testing
- ▶ System integration testing
- ▶ Performance testing
- ▶ User acceptance testing

We also consider coding standards and best practices as well as the creation of test cases and specific tools that might be used for testing.

The focus of Chapter 7, “Packaging and deployment” on page 201 is assembly, management and deployment of a store on the WebSphere Commerce Business Edition Version 5.4 platform. We present a practical example of packaging a store and deploying. We also address a method for making incremental changes to a store over its lifetime.

Part 1, “Customization examples” on page 215 presents examples extending the WebSphere Commerce Business Edition functionality. The approach taken here is to use real-world situations that are specific in their usage, but general in their applicability. The goal is that the lessons learned from these examples can be applied in many different situations.

Chapter 8, “Examples overview” on page 217 is an introduction to the second part of the book. In it we familiarize the reader with the sample stores in the chapters that follow. A summary of the functionality of each of the examples is provided as we explain how to work with them.

The first customization example is found in Chapter 9, “Orders” on page 233. Order processing is among the most basic aspects of a commerce system. It is also one of the most frequently altered as each site may have different requirements. There are many disparate scenarios that may exist in a commerce architecture. For example, a WebSphere Commerce deployment may be required to interface with a legacy ordering environment, integrate with an enterprise resource planning (ERP) system or process multiple repeating orders simultaneously. In this chapter, we focus on a couple of customizations: quick repeatable orders with an eye towards return customers and order integration with a back-end legacy system.

Chapter 10, “Shipping and taxes” on page 259 examines the extension of the built-in shipping and taxation functionality in WebSphere Commerce Business Edition. These topics become increasingly important as a company extends its reach into new markets. Globalization is adding new complications to tax calculations and shipping considerations. The first thing we look at here is customizing a store to build on the ability to ship by weight. We examine what is involved with calculating and adding the proper shipping costs based on location and range. We then turn our attention to taxation policies to explain the display of products with tax calculated in the displayed price. Finally, we discuss various aspects of discounts.

Chapter 11, “Messaging customization” on page 291 considers the inclusion of IBM’s MQSeries product family in a commerce site architecture. In the example, we focus on the implementation of an inbound message transaction. We show how to process an external input to the WebSphere Commerce environment via MQSeries resulting in updates to the database. Specifically, we examine the creation of a product via an inbound MQSeries message.

Chapter 12, “B2B features” on page 325 concentrates on the business-to-business shopping flow. B2B transactions is an area of online commerce that has taken on a life of its own in recent years. As the number of online vertical and horizontal partnerships continues to explode, the importance of integrating as seamlessly as possible with these partners will only increase. B2B is one of the strengths of WebSphere Commerce Business Edition Version 5.4, with many new and improved features supporting this model. In this chapter, we look to extend the basic B2B functionality included with the product. Specifically, we examine the improvement of the flow of a sample store to cater

to corporate buyers. The chapter is centered upon the concepts of working with contracts and catalog restrictions, role-based authentication and authorization and cost-based management approval wherein a transaction over a certain amount needs to be approved by another party prior to processing.

In Chapter 13, “Product entry and display” on page 349 we examine the basic display of the online catalog. We extend the functionality of WebSphere Commerce to improve the product comparison capability in such a way that allows a user to more easily find similar items and compare their attributes. We also present an approach to creating bundles of products as well as simultaneously displaying products in a catalog in more than one currency.

Chapter 14, “Migration examples” on page 397 moves the discussion to things that need to be considered in the case of a migration from an older WebSphere Commerce platform. We discuss any required shifts in methodology and code and examine the transition of old commands to new.



Architecture and programming model

This chapter provides an overview of the WebSphere Commerce architecture and programming model, with a summary of all the information you need in order to be able to design and develop a store.

It includes the following sections:

- ▶ Store architecture overview, containing an overview of the components of an online store, a brief description of the actual WebSphere Commerce store architecture and some samples of store configurations.
- ▶ WebSphere Commerce application architecture, containing a description of the application architecture from a business point of view, with a brief summary of its customizable components, subsystems and site assets.
- ▶ Websphere Commerce run-time architecture, containing a description of the components of the run-time architecture and some topology configuration samples.
- ▶ Websphere Commerce programming model, containing an overview of the J2EE architecture and programming framework, a brief description of the model-view-controller design pattern implementation in the WebSphere Commerce framework and an overview of the persistent object model.

Note that this chapter is intended to contain only summary and overview information, and each section contains references to product documents, other redbooks and Web sites for more details.

2.1 Store architecture overview

This chapter is an overview of the WebSphere Commerce store components and architecture, with some samples of store configurations. For more details, refer to *IBM WebSphere Commerce Store Developer's Guide Version 5.4*.

2.1.1 Store assets

The components of an online store, also referred to as store assets, are:

- ▶ Store front. This is what your customers see in the browser. In the WebSphere Commerce programming model, store fronts are implemented by means of Web assets such as HTML and JavaServer files, style sheets, images, graphics and in general multi medial files. Store developers can use tools like WebSphere Commerce Studio and WebSphere Studio Application Developer (for more information, refer to Chapter 4, “Planning and development” on page 71).
- ▶ Back office. This is the business logic implementation and customized code that your customers don't see. In the WebSphere Commerce programming model, back offices are implemented following the model-view-controller design pattern and the EJB model (see 2.4.2, “Model-view-controller design pattern” on page 40 and 2.4.3, “Persistent Object Model Overview” on page 45). Store developers can use tools like VisualAge for Java and WebSphere Studio Application Developer (for more information about development tools, always refer to Chapter 4, “Planning and development” on page 71).
- ▶ Store data. These are your store data assets, such are product catalogs and orders. Store developers have the following option for developing and editing them (for details, see 2.2.2, “Data assets” on page 19):
 - Use Store Services, in order to publish the store or to change general store settings as well as tax and shipping settings.
 - Use command utilities of the WebSphere Commerce Loader package, in order to load large amount of data into the WebSphere Commerce database: such utilities can be used at the initial database loading, or in order to propagate data changes, especially in case of schema changes when they are the only option.
 - Use WebSphere Commerce Accelerator, in order to edit data already in the database.

- Use your database specific utilities to perform SQL queries.

Information about publishing store components can be found in Chapter 7, “Packaging and deployment” on page 201.

2.1.2 WebSphere Commerce store architecture

In order to run your online store with WebSphere Commerce, you need following components:

- ▶ WebSphere Commerce Server
- ▶ WebSphere Commerce instance
- ▶ store assets

The WebSphere Commerce Server contain one or more Web applications, called WebSphere Commerce instances, each one connected to its own database. An instance can hosts just one store, or multiple stores published in different store directories, which can share the same data asset and use independent store front and back office assets. Some possible configuration samples are described in the following section.

Store configuration samples

The above described WebSphere Commerce components can be configured in several ways. The following are some samples which are represented in Figure 2-1 on page 14:

1. One server hosting an instance with a single store
2. One server hosting an instance with multiple stores which share the same data assets but use independent store front and back office assets
3. One server hosting an instance with multiple stores which use a shared catalog, but are independent for the other data assets, as well as for the store front and back office assets

Attention: The number of stores is limited by the WebSphere Commerce license that you have purchased. In the license agreement you can find information about how to purchase additional entitlements.

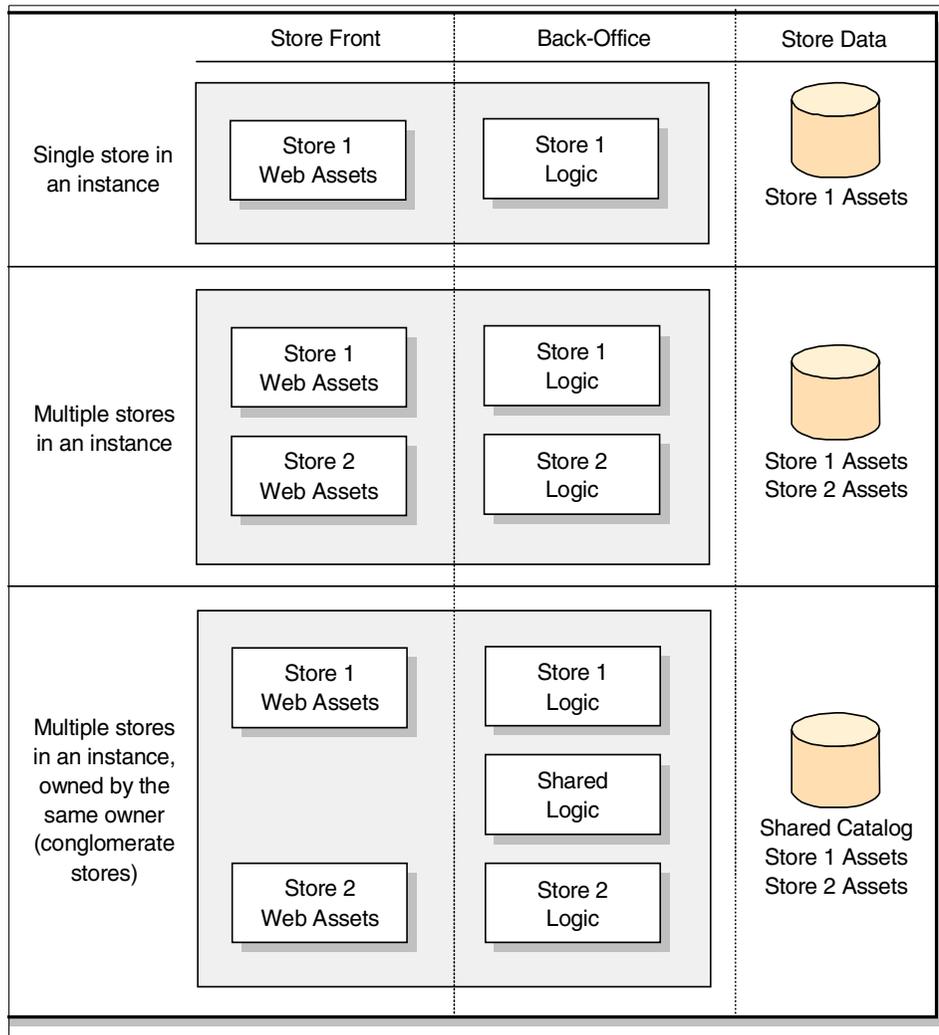


Figure 2-1 Store configuration samples

2.2 WebSphere Commerce application architecture

This section contains a description of the WebSphere Commerce application architecture and an overview of the components that can be customized for a store, including a description of the WebSphere Commerce server subsystems and a summary of the store data assets.

In the WebSphere Commerce manuals the application architecture is represented as in Figure 2-2 on page 16 (a) (refer to *IBM WebSphere Commerce Fundamental Version 5.4* and *IBM WebSphere Commerce Store Developer's Guide Version 5.4*). However, in this redbook it is tried a slightly different approach in order to achieve a more clear comprehension of the architectural concepts and components (see in Figure 2-2 on page 16 (b)). It follows a description of such concepts and components, with references to both the diagrams and explanation of the reasons behind the changes. Moreover, in the description is followed a top-down approach which seems more appropriate than the bottom-up approach adopted by the product manuals, because it is more consistent with the natural application development and run-time cycle.

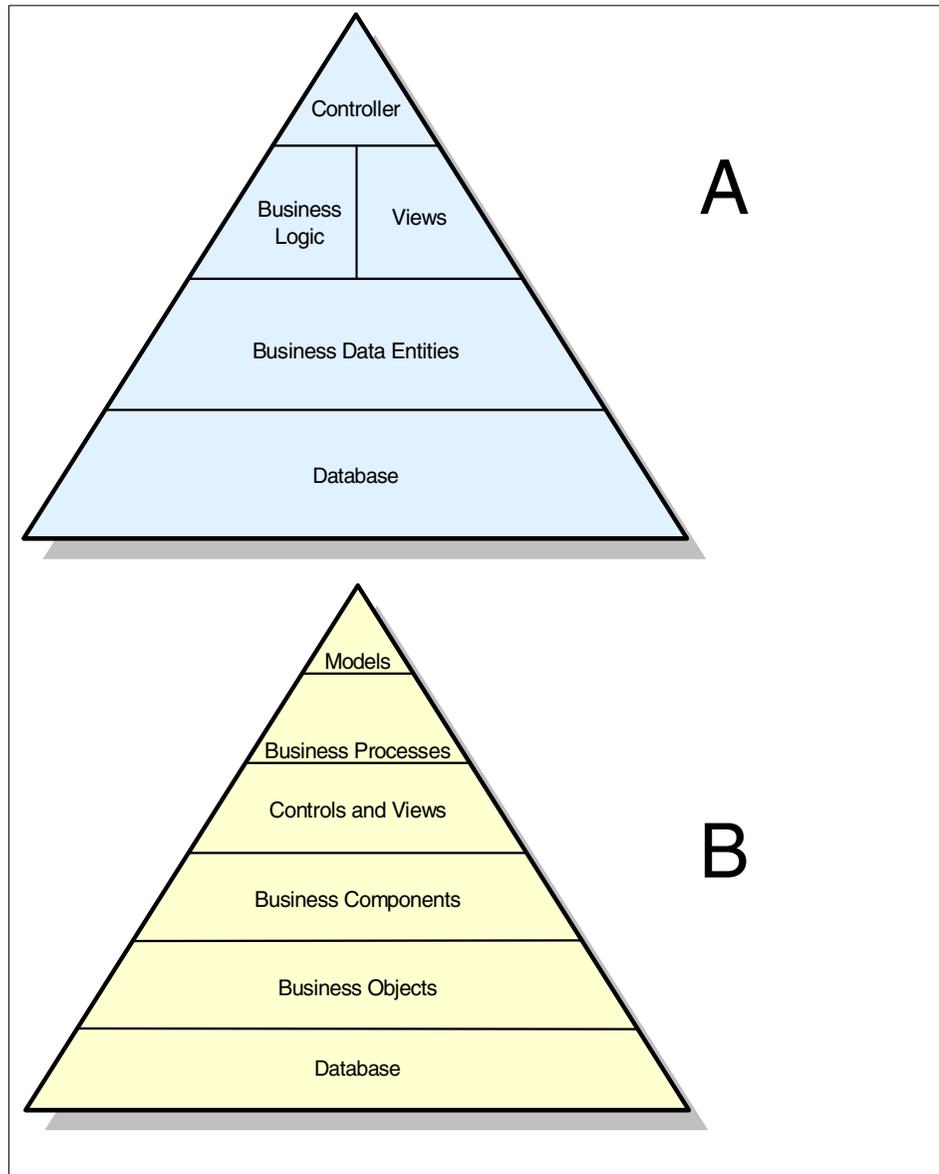


Figure 2-2 WebSphere Commerce application architecture (a) classical picture, (b) modified picture.

- ▶ Business models. A WebSphere Commerce application is based on e-commerce business models: the product provides sample stores for the most common models, such as InFashion, WebFashion and NewFashion for the business-to-consumer model, and ToolTech for the business-to-business

model. In diagram B of Figure 2-2, models occupy the top of the pyramid: they have been removed in diagram A of Figure 2-2 in order not to mix concepts of application architecture with actual implementation components.

- ▶ Business processes. Business processes, which occupy the second layer in diagram B are the workflow and site flow through which a store implements the adopted model. They have also they have been removed in diagram A of Figure 2-2 in order not to mix concepts of application architecture with actual implementation components. Samples of business processes are the Catalog Navigation and the User Registration processes, which can be applied to both the business-to-consumer and business-to-business models.

Next we describe the components that implement the architecture of WebSphere Commerce applications. These components are represented as layers of the pyramid shown in diagram A of Figure 2-2. They are listed starting from the top of the pyramid, so we can continue to follow the top-down approach already adopted:

- ▶ Web Controller. This is the component that handles the received service requests: for each request, it activates the appropriate business logic and invokes the appropriate view to create and send the response, consulting the WebSphere Commerce database registries. In diagram B of Figure 2-2 it is included in the Control and Views section.
- ▶ Business Logic and Views: These share the same level in diagram B of the Figure 2-2 pyramid because they interact with the same upper and lower levels. For more details about their implementation model, see “2.4.2, “Model-view-controller design pattern” on page 40”)
 - Business logic is split into units implemented as commands. WebSphere Commerce provides two types of business logic commands: controller and task commands. Controller commands encapsulate all the logic necessary to accomplish a single service request, such as OrderProcess for an order processing request. They use task commands to perform single units of work, such as payment processing in the previous example. Commands are represented as the Business Objects layer in diagram B of Figure 2-2.
 - Views are commands used to display in a browser the business logic results by means of an appropriate JSP template. They are part of the Controls and Views layer in diagram B of Figure 2-2.
- ▶ Business Data Entities. They contain the business data-centric logic. WebSphere Commerce implements them as Enterprise JavaBeans (V1.0), and provides access beans and data beans for an easier interaction with them:
 - Enterprise JavaBeans are the actual application interface to the database: they contain the logic to retrieve and modify data stored in it.

- Access beans provide commands with business data, hiding them the complexity of their physical format. They rely on the related Enterprise JavaBeans.
- Data beans are access beans extensions made with the purpose of providing views with simple business data containers.
- ▶ Database. This is the physical storage of the e-commerce application data. Typical tables of a WebSphere Commerce schema are:
 - Product
 - Order
 - User

Customizable components

Based on the descriptions given in 2.2, “WebSphere Commerce application architecture” on page 14, the following architecture components can be customized in WebSphere Commerce applications:

- ▶ Business logic can be extended and modified by
 - Adding new controller commands using new task commands
 - Making existing controller commands invoke new task commands
 - Extending already existing task commands.
- ▶ Views can be customized by adding new JavaServer Pages templates or modifying the existing ones.
- ▶ Business data entities can be extended with new Enterprise JavaBeans and their related databeans.
- ▶ Database. New tables can be added to the WebSphere Commerce database schema. For an overview of the existing tables, see 2.2.2, “Data assets” on page 19.

Attention: Avoid modifying the existing tables in order to simplify migrations to future releases.

2.2.1 WebSphere Commerce server subsystems

WebSphere Commerce Server is organized into the following logical subsystems, in such a way that the total customization required for an online store can be divided into a set of independent customizations, concerning one or more application components belonging to a subsystem:

- ▶ Catalog. It contains all the catalog business logic and data. It provides capability to navigate it, as well as to create and categorize products and

associate them in any way, including stock keeping units (SKUs), packages, bundles and merchandising associations. It also provides multicultural support.

- ▶ Order. It contains business logic and data for the order processing and management and any other related functions, such as taxation, inventory, payment and fulfillment.
- ▶ Member. It contains business logic and data related to members of the WebSphere Commerce system, such as users, group of users and organizational entity. It provides capability for members registration and profile management, as well as for authentication, access control and session management.
- ▶ Trading. It contains business logic and data related to the negotiation of price and quantity of product or set of products between buyers and sellers, including auctions support. In the WebSphere Commerce Business Edition, it provides also support for contracts and requests for quote (RFQs)
- ▶ Inventory. It contains business logic and data for real-time inventory management, including shipping and receiving inventory, as well as receiving inventory records both from vendors and, in case of returned items, from customers.
- ▶ Marketing. It contains marketing business logic and data, providing the capability to empower your store with marketing campaigns, product recommendations, advertisement, electronic coupons, discounts, customer profiling and collaborations.

For more information about WebSphere Commerce server subsystems, refer to Chapter 3 in *IBM WebSphere Commerce Store Developer's Guide Version 5.4*.

2.2.2 Data assets

The following sections contain a summary of the store data assets, represented in the WebSphere Commerce information model as shown in Figure 2-3 on page 20. More details about them can be found in *IBM WebSphere Commerce Store Developer's Guide Version 5.4*) and in the WebSphere Commerce Version 5.4 Online Help.

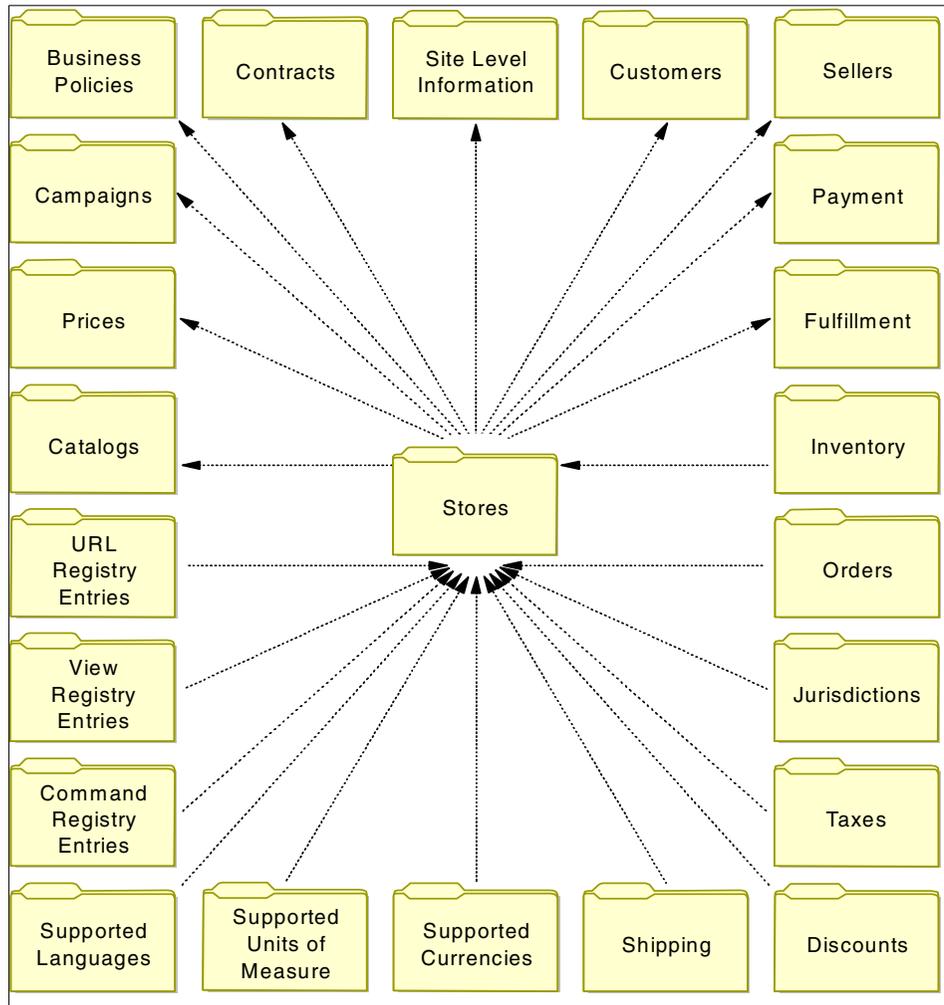


Figure 2-3 WebSphere Commerce store data assets

Site level info

The Site level component in Figure 2-3 represents the basic level of store data, which are stored into the database during the WebSphere Commerce Server instance creation, based on information provided with XML bootstrap files. Such data, that can be shared between multiple stores, includes:

- ▶ The default site administrator id, WCSADMIN.
- ▶ The default site organization
- ▶ The default organization to be used as store owner.
- ▶ The default URLs, commands and views.

- ▶ The default business policies
- ▶ Currency conversions and formats
- ▶ Device types and languages supported by the instance
- ▶ Message types and roles definitions (language-specific).

Stores

Core data is represented by the stores component in the middle of the diagram shown in Figure 2-3: the arrows exiting from it point to assets that can be shared between multiple stores, whereas the arrows entering into it come from store-specific assets. The following is the minimum set of store core data:

- ▶ The store identifier
- ▶ The store address
- ▶ The store directory
- ▶ The default contract
- ▶ The contract database tables must contain:
 - The store identifier
 - The member identifier of the organization which owns the store
- ▶ The Web assets store directory, in the STORE table.
- ▶ The unique store nickname or identifier, in the STADDRESS table.

Configuration Data

The following configuration data, which can be set both at site and store level, provides Web controller with run-time information about the appropriate commands and views to be invoked in response of each possible service request:

- ▶ URL registry entries, which map URL requests into command invocations.
- ▶ Command registry entries, which basically provide the command implementation class.
- ▶ View registry entries, which map the view names returned by commands with view commands and related JSP templates, based both on the store id and on the requesting device type.

Managed data

The following is a list of managed data created by the sellers, with a read-only access for customers (note that store-specific assets have been specified):

- ▶ Business policies
- ▶ Contracts
- ▶ Customers

- ▶ Sellers
- ▶ Payment
- ▶ Fulfillment centers
- ▶ Jurisdictions (store-specific)
- ▶ Taxes (store-specific)
- ▶ Discounts (store-specific)
- ▶ Shipping (store-specific)
- ▶ Supported currencies (store-specific)
- ▶ Supported units of measure (store-specific)
- ▶ Supported languages (store-specific)
- ▶ Catalogs
- ▶ Prices
- ▶ Campaigns

Operational data

The following data can be created and changed, directly or indirectly, by interacting with the site: such interactions can be performed both by customers and by sellers (note that, in this last case, operational data resulting from a particular type of interaction can be customers themselves):

- ▶ Customers
- ▶ Fulfillment
- ▶ Inventory (store-specific)
- ▶ Orders (store-specific)

Table 2-1 provides the list of tools which can be used to create and modify each type of data asset. For more information, refer to the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*) and the WebSphere Commerce Version 5.4 Online Help.

Table 2-1 Tools for data asset customizations

Tools for customizing data assets	Core data	Configuratio n data	Managed data	Operational data
WebSphere Commerce Loader package (data are loaded in form of an XML file).	Applicable.	Applicable.	Applicable.	In general, not applicable.

Tools for customizing data assets	Core data	Configuration data	Managed data	Operational data
Store Services	Store Services automatically build core data when a new store archive is created.	Not applicable	Only some of the managed data can be created and edited through the Store Services. See the WebSphere Commerce Version 5.4 Online Help for more details.	Not applicable
WebSphere Commerce administration console	The console allows to create the organization which has the role of store owner.	Not applicable	Not applicable	Not applicable
WebSphere Commerce Accelerator	Not applicable	Not applicable	The accelerator allows to create and edit the following data: <ul style="list-style-type: none"> ▶ Campaigns ▶ Contracts ▶ Fulfillment ▶ Discounts ▶ Catalog ▶ Prices 	The accelerator allows to create operational data for a customer, such as orders, and to manage the inventory.
Organization administration console (Business Edition)	Not applicable	Not applicable	Not applicable	The organization administration console allows you to create and approve buyers.

2.3 WebSphere Commerce runtime architecture

This chapter describes how the WebSphere Commerce application architecture, which has been presented in the previous section, behaves at run-time: it contains a summary of the run-time components and explains how they interact to accomplish a service request. Moreover, some configurations samples are presented. More details on the server run-time components can be found in *IBM WebSphere Commerce Programmer's Guide Version 5.4* and in the WebSphere Commerce Version 5.4 Online Help.

2.3.1 WebSphere Commerce runtime components

The WebSphere Commerce server run-time is a framework based on the Java 2 Platform, Enterprise Edition (J2EE) architecture (for more details about J2EE, refer to 2.4.1, “The J2EE Programming Platform” on page 40). Such framework provides the capability to handle system and user requests, and to perform the corresponding actions in order to fulfill them.

In the following, it will be given an overview of the characteristics of the server run-time components and a description of their interactions, as represented in Figure 2-4 on page 29:

- ▶ For http requests:
 - Servlet engine. It is the component of the WebSphere Application Server that handles http requests for WebSphere Commerce services: it receives the request from the WebSphere Application Server component plugged into the Web server, search into its pool of threads in order to find one to be dedicated to the request execution, and dispatch the request to the appropriate protocol listener.
- ▶ For non-http requests:
 - Transports. For non-http request, transports handle inbound and outbound messages of the assigned types, and dispatch them to the appropriate listener based on the request protocol. They are components of the messaging subsystem that can be configured through the WebSphere Commerce administration console. The pre-defined transports in the WebSphere Commerce sample stores are:
 - E-mail sender
 - File writer
 - MQ Series

Note that the messaging subsystem uses a Common Connector Framework (CCF) which allow to interact with the various transports through a common interface. Using this common interface it is possible to plug into the WebSphere Commerce messaging subsystem additional transports, in order to interact with enterprise resources such as CICS(R) and Encina(R) transactions. Samples about how to use the WebSphere Commerce messaging subsystem and how to implement back-end integration can be found respectively in Chapter 11, “Messaging customization” on page 291 and Chapter 9, “Orders” on page 233. For general information, refer to the WebSphere Commerce Version 5.4 Online Help or to the *WebSphere Commerce V5.4 Handbook Architecture and Integration Guide*, SG24-6567.

- ▶ Protocol listeners. They receive inbound requests based on the protocols and dispatch them to the adapter manager to find out the more appropriate adapter for the type of the requesting device. The most common protocol listeners are:
 - Request servlet. It is a servlet configured through the instance_name.xml file which handles incoming http requests.
 - MQ Connector, also called MQ Listener. It is a connector which uses the Common Connector Framework (CCF) and the Java Message Service interfaces in order to retrieve MQ Series inbound messages.
- ▶ Instance_name.xml file. It is the configuration file for WebSphere Commerce instances. The request servlet read it during its initialization, in order to find information such as adapters which are supported by the instance and must be initialized. Whenever possible, it should be modified using the Configuration Manager, but in order to set parameters of some components it must be directly edited (refer to the specific component’s documentation for details).
- ▶ Adapter Manager. It receives requests from the protocol manager and determines which among the available adapters is the most suitable one to process them, based on the type of the requesting device.
- ▶ Adapters. These device-specific components enable WebSphere Commerce with the capability to accept and fulfill service requests in XML format, coming from different types of devices, such as:
 - Standard internet browser.
 - Pervasive computing (PvC) devices, for instance personal digital assistants (PDAs) and cellular phones empowered with Internet browser: these devices send requests using wireless protocols, such as WAP and i-mode, which are converted into HTTP and HTTPS by a wireless protocol gateway before accessing the Web server.

- Back-end systems exchanging XML messages with WebSphere Commerce via MQ Series.
- WebSphere Commerce scheduler requesting the execution of background jobs: note that it does not require a protocol listener.

Adapters are responsible for the following tasks:

- Translate the message from the device-specific format into a WebSphere Commerce common format (i.e. a CommandProperty object).
- Instruct the Web Controller with the device requirements about the way to process the request.
- Provide device-specific session persistence.
- Determine the proper device format in which to generate the XML response: note that for the same view returned by the command, it is possible to specify into the View Registry different JSP templates for different requesting device types.
- Sending the XML response message.

Adapters provided by WebSphere Commerce are:

- Http browser adapter, for the standard Internet browser.
- Http PvC adapter, which is an abstract adapter for pervasive computing (PvC) devices. It has to be extended in order to support specific PvC applications.
- Program adapter, for remote programs invoking WebSphere Commerce commands by sending XML requests over the HTTP protocol.

Note that what is called the WebSphere Commerce MQ Series adapter, is actually a combination of the JMS-MQ connector which retrieves MQ messages and the program adapter which executes them.

- Scheduler adapter, for the WebSphere Commerce background jobs.

Note that the adapter framework can be customized and extended in the following ways:

- Develop a specific PvC device adapter.
- Create both a new protocol listener and a new adapter which implements the common DeviceFormatAdapter interface.
- ▶ Web controller. It's an application container that provides services such as:
 - session management, according to the session persistence established by the adapter
 - transaction control
 - authentication and access control

- enforcing the programming model, by requiring for example that controller commands always return view names.

The Web controller work flow is the following:

- Only for http request:
 - Start the transaction using the UserTransaction interface
 - Retrieve session data from the adapter
 - Whenever necessary, redirect the user to a logon URL
 - Whenever necessary, redirect the user to a HTTPS URL
 - Search into the URL registry the command interface corresponding to the request URL
- Execute the invoked controller command passing to it the command context and the input properties received by the adapter.
- In case of transaction rollback exceptions for a retrieable command, try to execute it again.
- After the command execution, check that a view name has been returned and look in the view registry for the view command to be invoked and, in case of forward view command, for the JSP templates to be used.
- Save and commit session data.
- Commit or rollback the current transaction, depending on success or failure cases.
- ▶ **Commands.** They are Java beans implementing the application business logic. WebSphere Commerce support four types of commands:
 - **Controller command.** It encapsulates the business logic necessary to fulfill a particular service request (in general an URL request), such as an order processing or an user registration. The controller command splits this logic in several units of works, each one performed by invoking a different task commands. Moreover, it is responsible to return to the Web controller a view name when the request has been successfully completed, and an error view name in case of exception.
 - **Task command.** It is invoked by the controller command to perform a single unit of work. It uses access beans to read and modify business data.
 - **Data bean command.** It is invoked by a JSP template in order to activate and populate a data bean.
 - **View command.** It handles the response to a client request in three possible ways:
 - Redirect view, to send the response using a redirect protocol.

- Direct View, to send directly the response.
 - Forward View, to forward the responsibility of the response to another runtime component, such as a JSP template.
- ▶ JavaServer Pages (JSPs) templates. They are JSP compliant files (see the following 2.4.2, “Model-view-controller design pattern” on page 40) which, after being compiled, become servlet specialized for display purposes. They are usually invoked by the Web controller after a command execution. They can also be invoked directly from the browser, but only if the request URL path includes the request servlet, which can perform the JSP execution in a single transaction, otherwise the data bean manager will reject the request in order to protect the JSP template (this is one of the possible access controls to WebSphere Commerce resources). JSPs templates can use data beans to retrieve business data, as described in the following items.
- ▶ Entity beans. They are Enterprise JavaBeans compliant objects (see the following 2.4.3, “Persistent Object Model Overview” on page 45) which model the WebSphere Commerce store data and contain the logic to retrieve and modify them, providing features such as persistency and transaction management. Moreover, they provide an interface to access the database schema without a direct knowledge of the physical characteristics of its tables. They can be customized either by extending existing entity beans or by developing and deploying entirely new ones.
- ▶ Access beans and data beans. They are useful and simple interfaces which hide the complexity of interacting with EJBs to task commands and JavaServer Pages templates. In particular, data beans extend access beans in order to be used as simple data containers by JSPs templates.

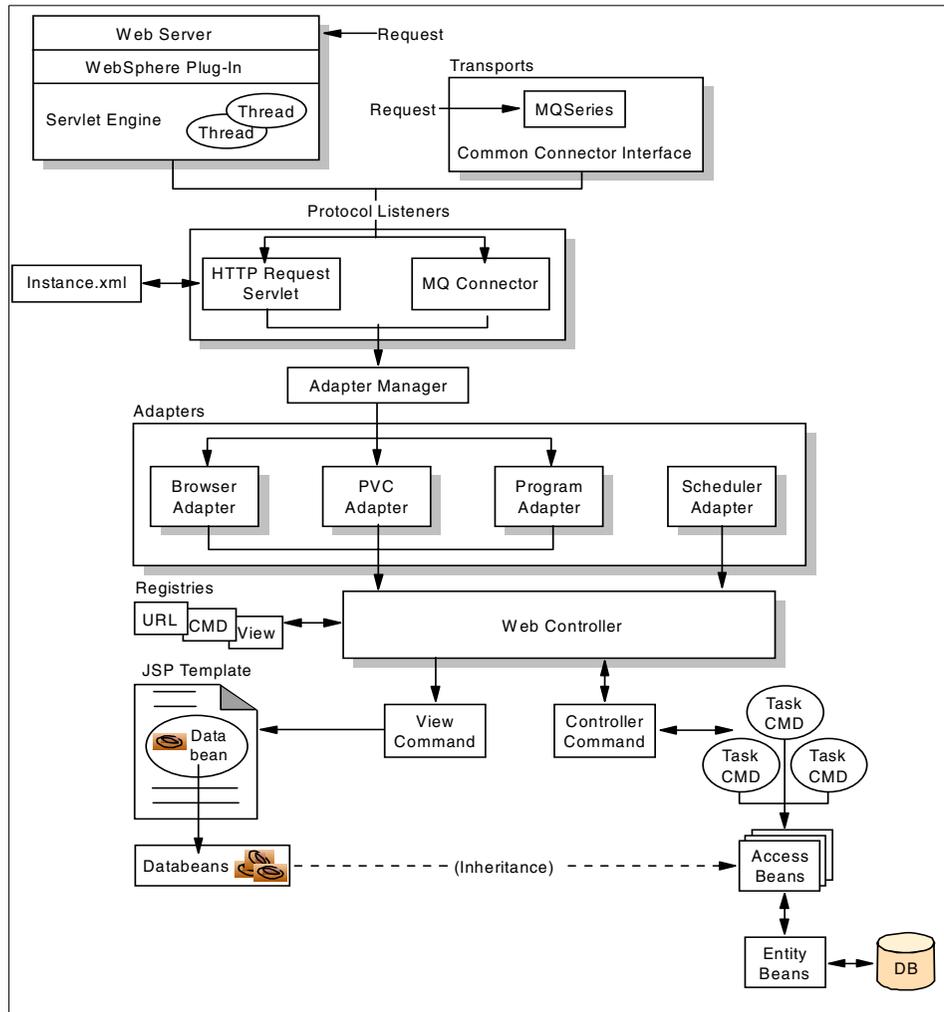


Figure 2-4 WebSphere Commerce run-time components.

Http request flow summary

This section explains step-by-step what happens when the WebSphere Commerce server receives an http request from an Internet browser, as represented in Figure 2-5:

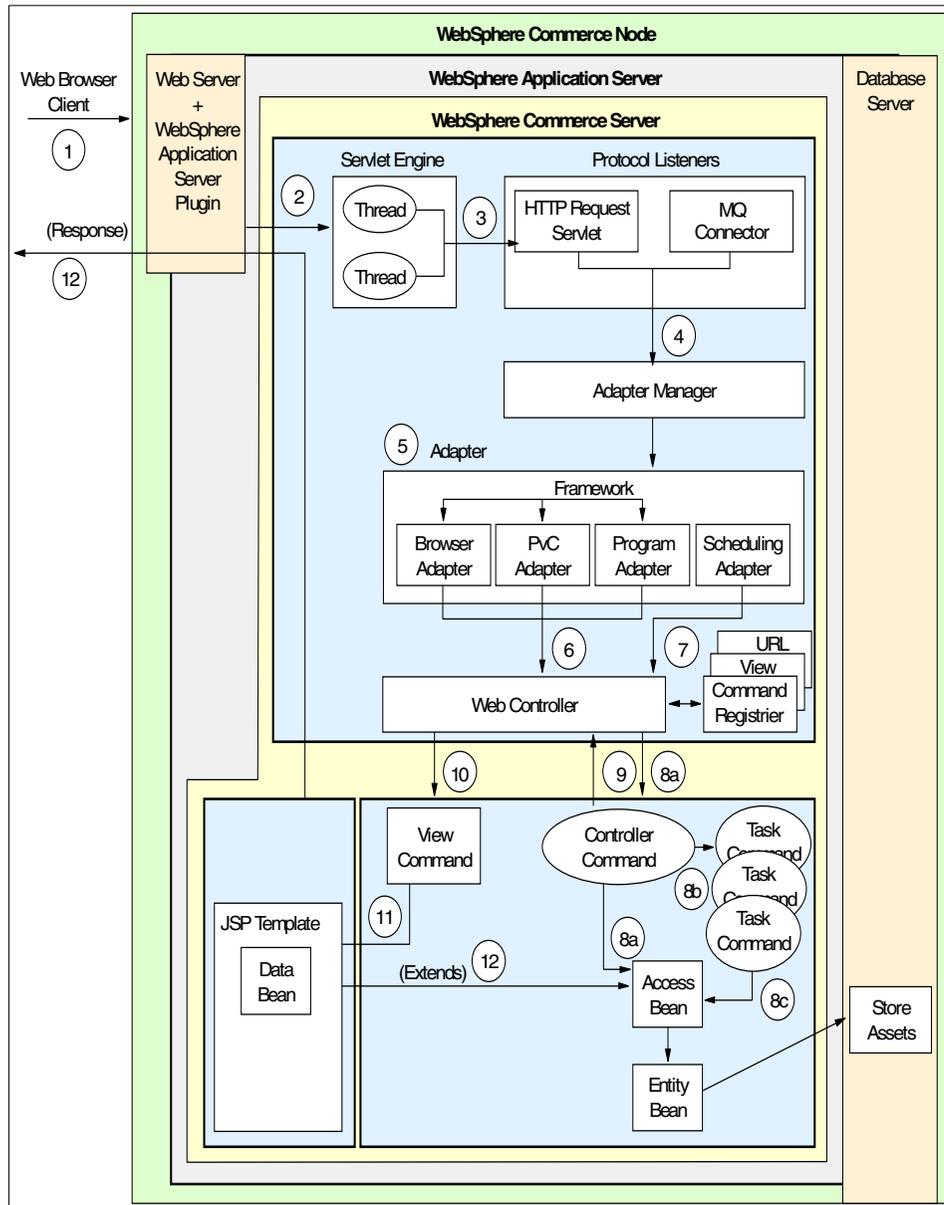


Figure 2-5 Http request flow in .

1. The client sends a request from a standard Internet browser using the Http protocol.
2. The WebSphere Application Server plug-in, running into the Web server, redirects the request to the WebSphere Commerce servlet engine.

3. The servlet engine gets a thread from its threads pool and dedicate it to the request processing. It dispatches the request to the http protocol listener, that is the request servlet.
4. The request servlet sends the request to the adapter manager to find out the adapter the more appropriate for the requesting device type.
5. The adapter manager determines that the request came from an Internet browser, and passes the request to the http browser adapter.
6. The http browser adapter dispatches the request to the Web controller.
7. The Web controller looks both into the URL registry, for mapping the request URL with the command interface to be invoked, and into the command registry, to find out the actual command implementation class (all these information can be specified both at site and at store level).
8. Following the most common path, the store has assigned to a controller command the responsibility to fulfill the request:
 - a. the Web controller invokes the controller command implementation class.
 - b. The controller command relies on a set of task commands in order to perform single units of work
 - c. Each task commands can access business data through access beans, which hide them the complexity of their related entity beans.
9. The controller command returns a view name to the Web controller, which looks into the view registry for the view implementation command to invoke (again (7) in the diagram). Keeping to follow the most common path, it finds a forward view command with the name of the JSP template to use.
10. The Web controller invokes the view command passing to it the JSP template name.
11. The view command forwards the responsibility to create the response to the JSP template.
12. The JSP template retrieves business data using data beans, an extension of entity beans.
13. The generated response is sent back to the browser.

2.3.2 Topology configuration samples

This sections explains the most common topology configuration issues, and presents as samples some topology configurations which are described with more details in *WebSphere Commerce V5.4 Handbook Architecture and Integration Guide*, SG24-6567. For more information about how to select a topology configuration, refer also to *IBM WebSphere Commerce Fundamental Version 5.4*.

General considerations

The architecture and configuration design of your e-commerce system must be the result of the technical strategy which has been defined in order to fulfill the business objectives and requirements of your store.

The most common focus areas for a technical strategy definition are:

- ▶ **Scalability.** Even if it is fundamental to make an estimation of what the number of users, the connection rate and load will be before starting to design and develop your Web site, it is usually difficult to be accurate at any degree in predictions involving the Internet world: for this reason, it is important to choose a configuration which allows your system to scale as the load of the incoming service requests increases, ideally for any reached size, simply by adding the appropriate number of resources or machines. The scalability of a system can be implemented in two ways:
 - Vertical scaling is implemented by adding resources such as processor, memory and software applications, on a single machine or node, in order to leverage all the available processing power.
 - Horizontal scaling is implemented by distributing multiple processes across multiple physical machines. It provides both increased throughput and failover (i.e. availability).

The cloning mechanism can be used for both the scalability implementations: it consists in creating multiple copies, identically configured, of an object, such as an application server, both on a single machine (multiple Java Virtual Machine processes) or on multiple physical machines. Moreover, it provides a simplified system administration as well as the possibility to perform workload management and to improve the system availability and reliability (see next items).

- ▶ **Performance.** Workload management is a mechanism that significantly improves the performance of your system by making each machine or server share a portion of the overall load that is proportional to its processing power. In this configuration it is fundamental to implement session management in order to maintain a session state between HTTP client requests (remember that HTTP is a stateless protocol that doesn't implement session management by itself). IBM Network Dispatcher is a separately orderable product that empowers WebSphere Commerce with workload management features. It manages Web servers grouped in a cluster by providing scalability, configurable load balancing and failover among them. Note that the client will appear to communicate always with a Web server, without having knowledge that it is actually the Network Dispatcher that intercepts and redirects requests. Network Dispatcher should normally run on a dedicated machine, and it is also possible to distribute it across two machines, in such a way to provide the system with dispatcher failover (for an

example of configuration including Network Dispatcher, see “3-tiers enterprise configuration”).

- ▶ Availability and Reliability. Online stores always require an availability as high as possible: in order to meet this requirement, single points of failure must be avoided by providing the run-time architecture of a failover system, that is hot-backup clones for each component.
- ▶ Security. Store assets and internal network of an e-commerce system must be always protected by potential intrusions by providing both operating system and network security. The level of protection to be applied to each run-time application component depends on the confidential data stored into the node and on the critical services provided by it. Each level of security can be implemented by adding a new firewall which perform access control from a less trusted network to a more trusted one. The most common types of firewalls are:
 - Screening routers, which are protocol firewalls usually used as a first layer of protection from the internet.
 - Application gateways, which are domain firewalls usually used as a further layer of protection before accessing the internal network (intranet).
Machines behind such firewalls are not visible from the internet.

The network area between the internet and the intranet is called Demilitarized Zone (DMZ): it is considered an high-risk zone, where it should be avoided to store confidential data, because it is protected only by a protocol firewall from the potential attacks coming from the internet. It contains one or more domain firewalls, depending on the implemented level of security.

The definition of a technical strategy must include also considerations about the project resources constrains, in order to define the more appropriate way to implement the above listed requirements based on a realistic financial investment.

The possible WebSphere Commerce configurations have the following constraints:

- ▶ The Web server, WebSphere Commerce Server, and database server must use the same operating system, even though they run on different physical machines.
- ▶ All the Web servers involved in the architecture of your subsystem must be of the same type (e.g. they must be all IBM HTTP Servers).
- ▶ WebSphere Commerce Analyzer or WebSphere Catalog Manager should better run on dedicated machines, in order not to compete with WebSphere Commerce server for system resources.

The following sections are dedicated to present some samples of topology configurations and to describe how scalability, performance, availability and security requirements are met by each one.

Single-tier configuration

A single-tier configuration consists of a single machine where all the WebSphere Commerce run-time components run. It is the most easy and less expensive configuration to set-up and maintain: it is ideal for self-contained development and test environments and can be used for small stores with moderate transaction volumes. In all the other cases should be avoided, unless you're using an IBM zSeries or iSeries server which has its own implementation of the scalability, performance and availability requirements.

Scalability. It is the only case where the system can be scaled only vertically, by adding processors, physical disk space and memory to the single machine.

Performance. Only moderate loads are supported because application server and database compete for the same system resources.

Availability. The whole system has a single point of failure.

Security. Only the protocol firewall can be implemented.

A sample of single-tier configuration is shown in Figure 2-6 on page 35.

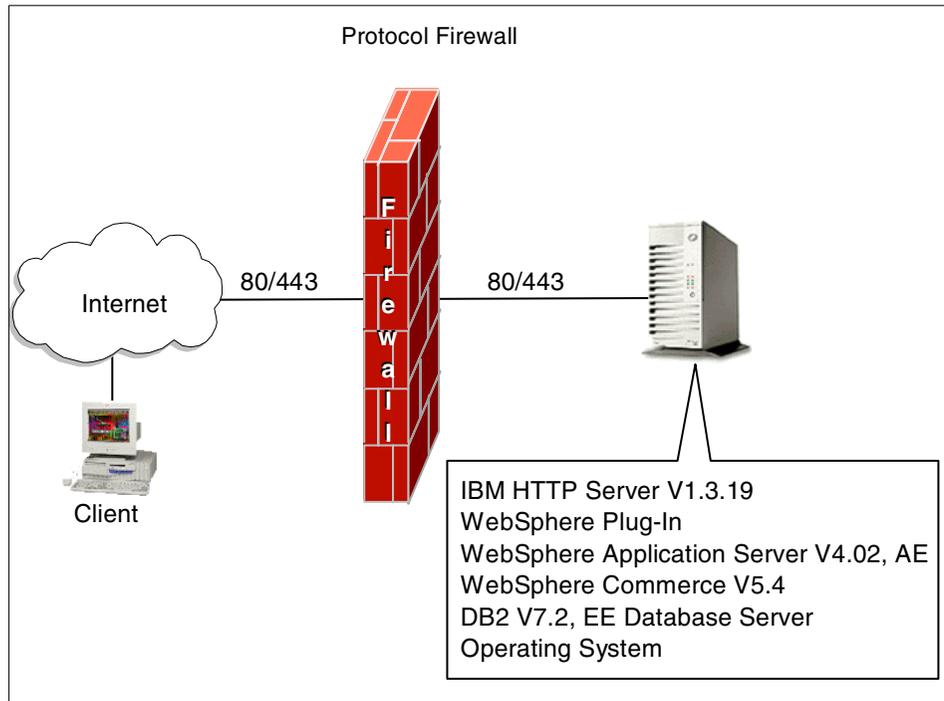


Figure 2-6 WebSphere Commerce single-tier configuration.

2-tiers configuration

In a 2-tiers configuration the database server run on a different node, usually behind a domain firewall. This configuration can be adopted by medium-to-large store with transaction volumes higher than in the single-tier case: in fact, it gives more possibilities by allowing a separate tuning and configuration of the Web and database servers. Its limit is that WebSphere Commerce server must run on the same node of either the Web server or the database server.

Scalability. Higher scalability can be reached by scaling separately the Web and database servers: in particular, it can be adopted a multiple Web server configuration consisting of multiple nodes with Web and WebSphere Commerce servers all accessing to a common database.

Performance. A better tuning can be performed by configuring separately the Web and database servers. In particular, in case of multiple Web server configuration, a Network Dispatcher can be used in order to provide an effective workload balancing.

Availability. An high availability strategy can be implemented by adopting a multiple Web server configuration together with a database failover system.

Security. A domain firewall can be put in front of the database, and it can be configured in order to allow access only to the WebSphere Commerce server, in case it run on the same node as the Web server. In this last case, a further level of security cannot be inserted between WebSphere Commerce server and the Web server.

A sample of 2-tiers configuration is shown in Figure 2-7.

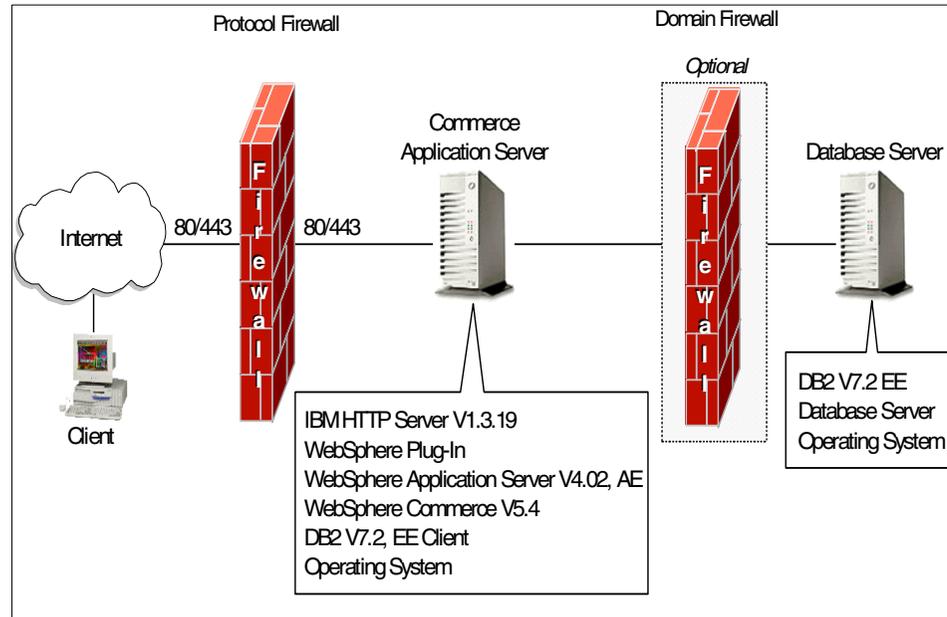


Figure 2-7 WebSphere Commerce 2-tier configuration

3-tiers configuration

In the 3-tiers configuration Web server, WebSphere Commerce and database server run on different nodes. This configuration allows to implement a further level of security, by setting up a demilitarized zone (DMZ) where only the Web server run, and provides almost unlimited expansion possibilities. It is recommended for large enterprises whose stores have high transaction volumes and security issues.

Scalability. It is highly improved by the possibility to scale separately the Web server, WebSphere Commerce server and database server components.

Performance. This configuration provides the highest degree of flexibility in performance tuning by allowing to configure separately Web, WebSphere Commerce and database servers. Moreover, it gives the possibility to implement the most effective workload balancing by means of one or more Network Dispatchers.

Availability. Single points of failure can be avoided at all by implementing both a redundancy strategy of multiple and integrated Web and WebSphere Commerce servers and a database failover configuration.

Security. In this configuration only Web servers must run in the DMZ and be visible from the Internet, whereas WebSphere Commerce and database servers can be protected by implementing one or more domain firewalls.

A sample of 3-tiers configuration is shown in Figure 2-8.

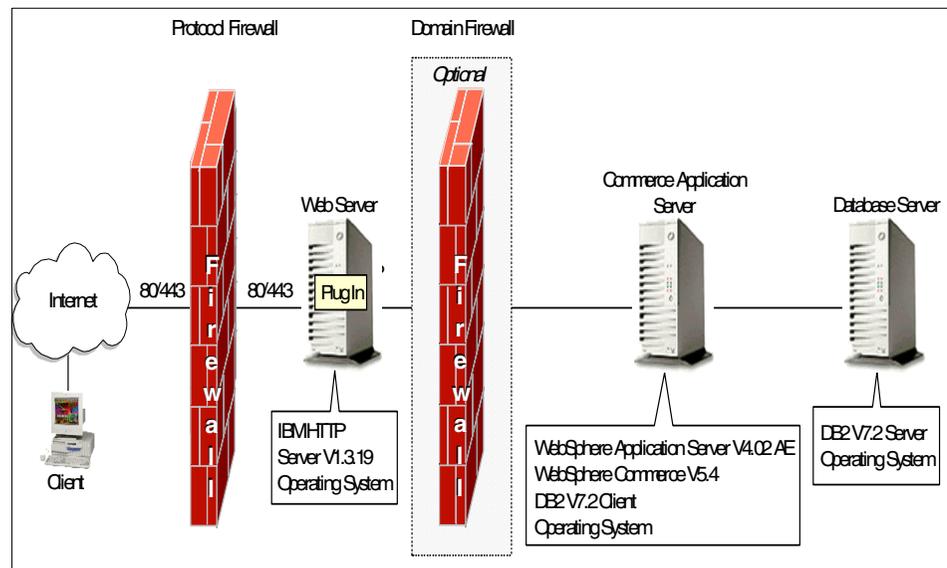


Figure 2-8 WebSphere Commerce 3-tier configuration

3-tiers enterprise configuration

A possible implementation of a 3-tier configuration for an enterprise is represented in Figure 2-8.

2.4 WebSphere Commerce programming model

WebSphere Commerce instances run as Web applications hosted by WebSphere Application Server, which provides them with an infrastructure and a component model fully compliant with the Java 2 Platform, Enterprise Edition (J2EE) standard.

After a brief overview of the Java 2 Platform, Enterprise Edition, the following sections describe how the WebSphere Commerce programming model leverages the J2EE compliant components and infrastructure provided by WebSphere Application Server.

The high-level information contained in this chapter can be integrated by the following

- ▶ Product manual:
 - *IBM WebSphere Commerce Programmer's Guide Version 5.4*
- ▶ Other redbooks:
 - *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167
 - *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176
 - *WebSphere Version 4 Application Development Handbook*, SG24-6134
- ▶ Articles:
 - Possible approaches to follow when designing a J2EE e-commerce solution are described in “*Approaches for e-commerce architectures - A conceptual guide for using J2EE in your design*”, downloadable at the URL <http://www-106.ibm.com/developerworks/library/j-ecomm/index.html>
 - A step-by-step process to develop your first J2EE application is described in the article “*Developing and Testing a Complete "Hello World" J2EE Application with WebSphere Studio Application Developer beta*”, downloadable at the URL http://www7b.boulder.ibm.com/wsdd/techjournal/0110_wosnick/wosnick.html
- ▶ Useful Web sites to keep among the browser's Favorites:
 - The sun site <http://java.sun.com/j2ee> contains references to articles, tutorials and the complete version of the Java 2 Platform, Enterprise Edition specifications.
 - The Java technologies section of the IBM developer works site, <http://www-106.ibm.com/developerworks/java/> provides articles, forums and information specific for developers who works with IBM products.

2.4.1 The J2EE Programming Platform

Java 2 Platform, Enterprise Edition (from here on, referred to as J2EE) is a set of specifications integrated and coordinated in order to provide a development platform for distributed and object oriented enterprise applications. Note that it does provide a component model and a run-time infrastructure, but it doesn't specify an architecture for enterprise applications, leaving architects free to design the best solution that meet their business requirements.

The main advantages that can be obtained by adopting the J2EE programming platform are:

- ▶ Shorter development cycles. As an object-oriented language, Java allows to distribute the business logic in self-contained and reusable components. Moreover, developers can focus only on implementing the application specific functions, by leveraging the infrastructure services provided by the so-called J2EE “containers”.
- ▶ Maintainability. Self-contained and reusable objects make J2EE applications much more easy to maintain.
- ▶ Portability. J2EE applications can run on several platforms, giving you, for example, the possibility to have different development and production environments.
- ▶ Connectivity. The J2EE technology provides applications with a powerful infrastructure that allows enterprises to connect the already implemented business logic of their back-end systems with Web browsers as well as pervasive computing (PvC) devices, such as personal digital assistants (PDAs) and cellular phones.

2.4.2 Model-view-controller design pattern

As already stated, one of the most important requirements for enterprise applications in general, and for online stores in particular, is the ability to fulfill service requests coming from different kinds of devices, each one characterized by a different type of user interface: HTML pages for standard Web browser, WML for pervasive computing (PvC) devices, and XML messages for business suppliers.

The model-view-controller architecture provides J2EE applications with the possibility to use always the same functions and data, independently by the final user interface, by keeping separate business and presentation logic. This separation, invented by Smalltalk and adopted by J2EE, makes supporting multiple types of clients much more easy to implement, test and maintain.

In particular, the model-view-controller components are:

- ▶ The model. It contains all the business data and functions, but does not have any knowledge about the clients requesting them. The architecture is frequently improved by implementing a further subdivision of the model into:
 - The domain model, consisting only of the business data and of the logic to access them. In J2EE applications, the domain model is usually implemented by Enterprise JavaBeans and data beans (see 2.4.3, “Persistent Object Model Overview” on page 45).
 - The application model, consisting of the business processes that manipulate such data. Note that the application model knows what views must be invoked by the controller in order to display the results of its processes, but it doesn’t know anything about the type of the requesting device. In J2EE applications, the application model is usually implemented by Java beans which follow the command pattern (see the following section “Command design pattern”).
- ▶ Views. They render data obtained by the model in the way the more suitable to the requesting devices. In J2EE applications, views are often implemented as JavaServer Pages templates producing output in HTML, WML or XML format.
- ▶ Controllers. They handle the application work flow by mapping service requests coming from user interfaces into actions to be performed by the model: when the results are available, they invoke the appropriate view in order to display them. In J2EE applications, controllers are usually implemented as servlets.

Figure 2-10 on page 42, taken from the *IBM WebSphere Commerce Programmer's Guide Version 5.4*, represents how the model-view-controller design pattern is implemented by the WebSphere Commerce architecture.

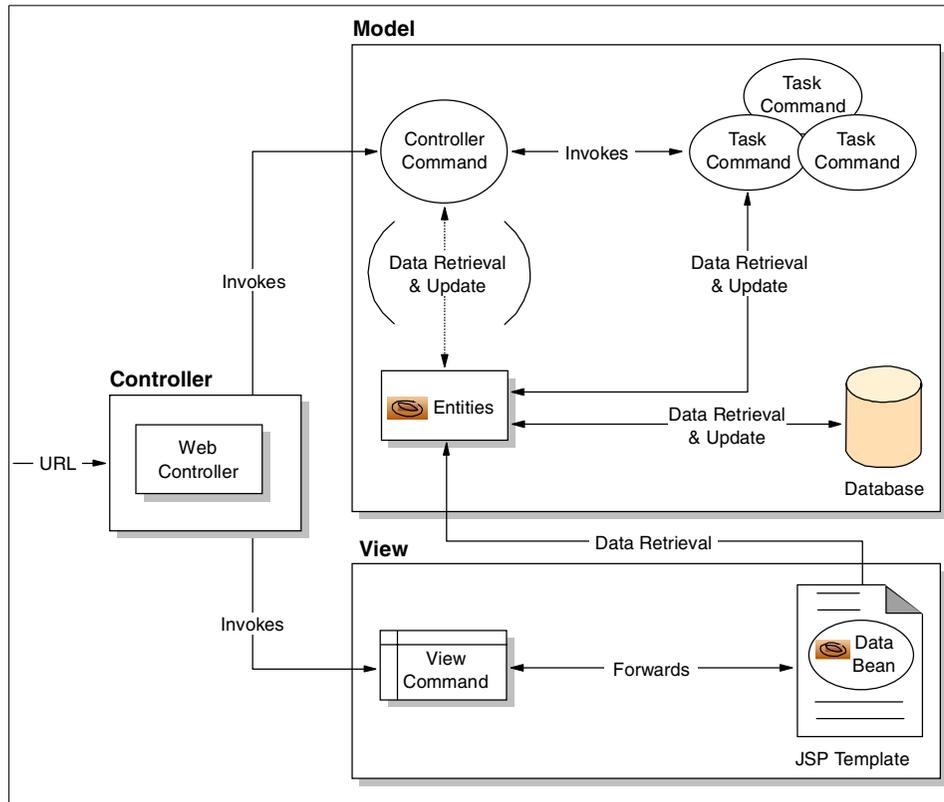


Figure 2-10 WebSphere Commerce implementation of the MVC design pattern.

The model-view-controller design pattern is the most famous example of the model-view-presenter programming model, described in “MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java“, an article downloadable at the URL

<ftp://www6.software.ibm.com/software/developer/library/mvp.pdf>.

Command design pattern

As already described in 2.3.1, “WebSphere Commerce runtime components” on page 24, service requests coming from different types of devices are translated by adapters into a common format that can be understood and processed by WebSphere Commerce commands. Moreover, the Web controller translates a service request into an actual command invocation, based on the type of the requesting device and on the addressed store id. Commands are objects that

follow the Java bean naming conventions. They perform some piece of business logic by leveraging the WebSphere Application Server command framework, which is an implementation of the standard command design pattern characterized by:

- ▶ Having both an interface and an implementation class for each command.
- ▶ Using a command factory in order to map the interface with the correct implementation class to be invoked, based both on the default command class name of the interface, and on the database command registries.
- ▶ Allowing clients to invoke commands only by using their interface and through the following sequence of invocation steps:
 - Set the command's input properties
 - Invoke an *execute()* method
 - Retrieving output properties.

WebSphere Commerce supports four different types of commands:

- ▶ Controller commands. They encapsulate all the logic necessary to accomplish a single service request, such as OrderProcess for an order processing request. They invoke task commands to perform single units of work, such as payment processing in the previous example, and control the application logic flow in order to fulfill the whole request. Upon completion of the flow, they return a view name to the Web controller, which is in charge to determine the view implementation class for the particular store and requesting device.

Controller commands are targetable, which means that they can be executed on a different container, but only the local target is supported.

- ▶ Task commands. Each task command performs a single unit of work, and usually access a single business data entity using an access bean wrapper that hides it the complexity of interacting with Enterprise JavaBeans (see 2.4.3, "Persistent Object Model Overview" on page 45). Task commands are not targetable, which means that they must be executed on the same container as the invoking controller command.
- ▶ Data bean commands. They are invoked by JavaServer Pages templates through the data bean manager in order to populate data beans. They are targetable, but it is supported only the local target.
- ▶ View commands. They can be invoked either by a controller command returning their interface name to the Web controller, or directly by a client. They are of three different types:
 - Redirect view command, which sends the view using redirect protocol.
 - Direct view command, which sends the view directly to the client.

- Forward view command, which forwards the view request to another Web controller, usually a JavaServer Page template. They are targetable, but it is supported only the local target.

The level of indirection in commands invocation that is provided by the WebSphere Application Server command framework, brings to the following consequences:

- ▶ The possibility of implementing an access control policy based on the requesting user at command level.
- ▶ The flexibility to execute different command implementations for different stores.
- ▶ The flexibility to execute different command implementations for different types of requesting devices.

The command flows has been presented both in Figure 2-4 on page 29 and Figure 2-10 on page 42.

Display design pattern

According to the display design patterns, view commands are implemented as JavaServer Pages templates, that is they are developed in the JSP script language which provides easily mechanism to access and display data without the necessity of a particular programming skill. At run-time, JavaServer Pages templates are compiled into servlets after the first invocation. They could be implemented directly as servlets, but this way is not recommended because their implementation would require a much higher level of programming skill.

Moreover, view commands should be invoked only by using their view name: as usual, this level of indirection gives the possibility to invoke at run-time the appropriate implementation. For example, different view implementations can be invoked based on the locale, the type of the requesting device or any other property into the request context.

JavaServer Pages templates use data beans, which are access bean extensions that allow them to retrieve business data in a very simple way. In fact, they can create and populate data beans by means of the data bean manager with a single line of code, like the following:

```
com.ibm.commerce.beans.DataBeanManager.activate(DataBean,HttpServletRequest)
```

Note that the above line of code is automatically created by WebSphere Studio Page Designer and WebSphere Studio Application Developer whenever a new data bean is insert in a JavaServer Page template.

After the activation, JavaServer Page templates can simply retrieve any property they need from the data bean, and even get the property names in all the languages supported by the WebSphere Commerce instance in case of multicultural stores.

Data beans can retrieve data in two different ways:

- ▶ Smart data beans retrieves associated data only when they are actually requested (lazy fetch): they provide better performance but they are more complicated to develop.
- ▶ Command data beans rely on a command in order to retrieve all their data at once: they are easier to implement but they can bring to higher performance costs.

2.4.3 Persistent Object Model Overview

WebSphere Commerce implements its persistent object model with entity beans compliant to the Enterprise JavaBeans specifications, V1.1.

Enterprise JavaBeans

Enterprise JavaBeans (EJBs) are standard wrappers of business data that are provided by the EJB container with the following services:

- ▶ Persistency, in case of either entity EJB or of stateful session EJB (see in the following).
- ▶ Distribution. They can be executed remotely by means of the Remote interface. Moreover, they can be created and discovered by network applications by means of their Home interface and their primary keys.
- ▶ Transaction management.
- ▶ Security implementation by means of access control mechanisms.

The main types of EJBs are:

- ▶ Entity EJBs persists until it is explicitly removed from its container by either the client or the server. Based on the type of persistence management, they can be
 - Container-managed-persistence (CMP), if they rely on the container for their persistency.
 - Bean-managed-persistence (BMP), if they implement their own persistency.
- ▶ Session EJBs persists only as long as either client or server maintains a reference to them. They can be:

- Stateless, when a single instance can handle requests from multiple clients, receiving all the necessary input parameters with each requests. This is the most scalable solution.
- Stateful, when and EJB instance maintains a state between invocations. Note that this solution impacts the scalability of the whole system, because it requires that a client is connected always with the server which hosts the EJB that it references.

For more information about Enterprise JavaBeans, refer to the “EJB Tutorial”, downloadable at the URL <http://www.ejbtut.com>

and to the article “*What are Enterprise JavaBeans components?*”, downloadable at the URL <http://www-106.ibm.com/developerworks/library/what-are-ejbs/part1/index.html>, which in three parts explains the goals of the EJB architecture, the programming model, and how to deploy and use EJBs.

WebSphere Commerce EJBs

WebSphere Commerce uses mainly CMP entity beans, with some exceptions of stateless session beans. Moreover, it provides CMP entity beans with an extension to the standard which consists in the possibility of establishing inheritance relationships between beans defined into the same container, called EJB group.

WebSphere Commerce includes two type of EJB groups: the private EJBs are used by the system and should not be extended or used by the application. On the other hand, the public EJB groups can be both used and customized. They are the following:

- ▶ WCSActrIEJBGroup
- ▶ WCSApproval
- ▶ WCSAuction
- ▶ WSCCatalog
- ▶ WCSCommon
- ▶ WCSContract
- ▶ WSCCoupon
- ▶ WCSFulfillment
- ▶ WCSInventory
- ▶ WCSMessageExtensions
- ▶ WCSOrder
- ▶ WCSOrderManagement
- ▶ WCSOrderStatus
- ▶ WCSPayment
- ▶ WCSPVCDevices
- ▶ WCSTaxation

- ▶ WCSUserTraffic
- ▶ WCSUser
- ▶ WCSUTF

The WebSphere Commerce persistence model is a framework that can be extended by:

- ▶ Extending an existing public entity bean.
- ▶ Adding new entity beans to an existing public EJB group.
- ▶ Adding both new entity beans in a new EJB group.

Important: Some of the public EJB groups contain session beans: customization of session beans should be avoided in order to simplify migrations to future releases.



Requirements and design

This chapter introduces a methodology that can be used for the development of your WebSphere Commerce site and shows the application of that methodology to the sample stores we create in the second part of this book. We examine the need for crisp requirements gathering and apply the requirements to the design phase of an engagement. Information about working directly with the customized stores themselves can be found in Chapter 8, “Examples overview” on page 217.

3.1 Application development methodology

The application development methodology we introduce below is designed for development efforts of medium to large complexity. This process can be used to tailor WebSphere Commerce sites that require a fair degree of customization.

The custom development approach, shown in Figure 3-1, is a proven component of the IBM Global Services Method, the preferred methodology for delivering business solutions for IBM and its customers. The phase-based process is designed to be repeatable over the lifetime of a solution and allows for extensibility within each phase.

There are 7 phases in the process: solution startup, solution outline, macro design, micro design, build cycle, deployment and solution close. Each phase is comprised of a set of activities that lead to the completion of the phase. As you can see from the diagram, the process becomes iterative during the release portion of the development process, allowing for a high degree of flexibility and change in your solutions, Let's take a closer look at the individual phases and their subcomponents.

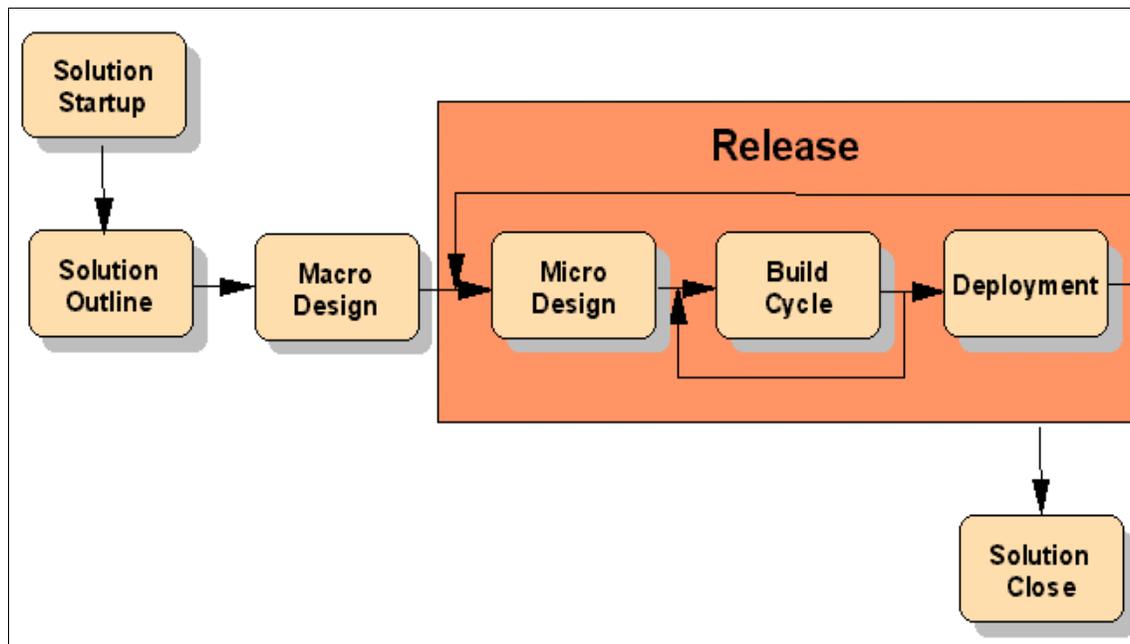


Figure 3-1 Custom development engagement process

3.1.1 Solution startup

The first phase in the custom development engagement process is the solution startup. Composed primarily of the tasks necessary to define and start an engagement, this phase includes clarifying and agreeing upon the scope of work, compiling detailed risk plans and create an overarching definition of the project.

Essentially a detailed project management exercise, this phase normally defines the team participation and organization. It also sets clear objectives for the solution and uses these objectives to create a finalized project plan. It serves to point out how important it is to have a crisp definition of the project at the outset. By having agreements set up at the beginning of a project, it is much simpler to handle customer expectations and scope creep.

3.1.2 Solution outline

The next phase, solution outline, is where the complexity and scope of a solution are determined. With a better understanding of the scope and complexity, the effort can be managed more appropriately.

Client environment

The first step in defining the outline is to gain an understanding of the client environment. The firmer your grasp on the customer's business and organization, the tighter a fit your solution will be. You should understand the customer's business processes, strategic goals and current architecture. This should then be documented as well as time permits for the benefit of the solution team.

Functional and non-functional requirements

The next step is to black box the solution and describe the requirements for it. What should this system be required to do? What are its scope and usage characteristics? Requirements gathering is a particularly important component of the process as it serves as the first blueprint that the solution team will be using to create the system.

It is here that functional and non-functional requirements are created as are use cases which can be used later in the testing activities. Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system will be required to perform. Some of the functional requirements to consider for a commerce system are:

Audience	To whom will the store cater? Is this to be a site for consumers or businesses? How will the site be accessed?
----------	--

Flow	How will the shopper be directed?
Catalog	What products or services will be sold?
Order	How will orders be processed?
Language	What languages will be supported?
Payment	What types of payment will be supported? Will multiple currencies be required?
Support	How will customers be supported during and after their experience?

Functional requirements are frequently described through the facility of use cases. A use case defines a goal-oriented set of interactions between external actors and the proposed system. It captures who does what with the system and why they do it, normally without describing the system internals. Some of the use cases we developed for the examples in the second part of this redbook can be found in Section 3.2, “Example store requirements” on page 57.

Non-functional requirements can be seen as constraints on various attributes of the functional requirements. They add a degree of clarification to the project by defining the system within the bounds of the customer’s environment. We can break non-functional aspects of a solution into nine major themes: performance, security, scalability, data integrity, availability, maintainability, manageability, environmental and system usability. Examples to consider for development of a commerce site include:

- ▶ Response time and throughput requirements
- ▶ Data retention and integrity throughout the commerce environment
- ▶ Uptime requirements
- ▶ Portability
- ▶ Integration with existing client systems
- ▶ Existing technical standards

The functional and non-functional requirements should be compiled with and agreed to by the client in order to avoid any problems during the delivery of the solution. More information on requirements and use cases in store planning can be found in *IBM WebSphere Commerce Store Developer’s Guide Version 5.4* and *IBM WebSphere Commerce Suite Fundamentals*.

Application and architecture models

Following the initial requirements definition activity, the application and architecture models should be outlined using the business requirements as input. The application model is a high-level draft of the application which provides a feel for the complexity of the commerce solution. It should show the major components of the solution and their relationships. Similarly, the architecture model is a high-level draft of the infrastructure needed to deploy the application. Both of these tasks are iterative and their output will likely be altered and refined during the course of the project.

Solution strategy

The solution strategy phase is where you assess the impact of the commerce solution to the business and develop a strategy for approaching the development and implementation of the solution. As in the outline of the application and architecture models, the requirements are an important component of this activity. The use cases should be grouped and ranked according to some set of criteria agreed to by the customer. From this a refined project plan can be devised that will serve as the outline for the remainder of the project.

Common considerations at this point in the project include the definition of:

Store usability design	Determine shopping experience characteristics and site design
Catalog management	Establish roles and processes for maintaining the catalog data
Deployment plans	Define the change management process by which application components are to be released
Store configuration	Set basic rules for how stores are going to be configured internally
Testing strategies	Define the high-level approach and activities surrounding the direction for testing the store

3.1.3 Macro design

As you can see from Figure 3-1, the release process, comprised of the micro design, build cycle and deployment phases, is designed to be a repeatable process. There are, however, things that span a number of releases; for example, certain aspects of the user interface, the architecture model and commerce test cases may remain static across several releases of the site.

The macro design phase is centered around preparing for the work involved with creating multiple releases. It addresses the architectural issues that are common across releases. It is obviously better to spend the time doing this work once up front rather than iterating through the same work for each release as this approach saves both time and money.

Steps that may be taken in this phase include:

- ▶ Develop a system-wide infrastructure component model. The basic infrastructure of a commerce site should not change greatly from release to release. This task outlines the core hardware and network infrastructure that will be used and how they interconnect. Should there be a change in an exiting function or a new function added, it can be handled in the micro design phase of a given release.
- ▶ Design the shopping flow. Working from the requirements gathered and processed in earlier phases, the shopping flow of the site can now be established and defined. This should be built on the work involving the type of store you will be creating, how users will be accessing the store, the nature of the goods or services that will be sold and what languages and currencies will be supported. This is also where you would define a consistent image for the site such that the user interface is consistent from release to release and brand management is preserved.
- ▶ Ensure the development environment is properly equipped and running. The tools that the development team needs to create the commerce site may include WebSphere Commerce Studio, VisualAge for Java and WebSphere Studio Application Developer. These tools, outlined in , need to be installed and operational for the development team to use over the course of multiple releases.

3.1.4 Micro design

Micro design is the first of three phases repeated for each release. Where previous phases have a solution-wide scope independent of scope, the micro design, build cycle and deployment phases drill down to the details of a specific release.

In the micro design phase, a specific release of the commerce site is clearly defined. This phase is essentially a refining of the solution based on all of the work that has been performed up to this point.

The first activity to perform here is a gap analysis of the needs of the release against the output from the macro design phase. This will point out any new requirements specific to this release. Next, the developers take the information gathered during the previous phases and apply it directly to a specific

implementation platform to create a physical application model. In our case, of course, the implementation platform is the WebSphere Commerce programming model. Information on the programming model can be found in Chapter 2, “Architecture and programming model” on page 11.

Micro design involves not only honing the use cases into very specific shopping flows and interfaces based on the needs of the release, but also describing the internal operation of objects and building the commerce application database schema.

3.1.5 Build cycle

The build cycle phase is where the solution is incrementally developed and tested until the objectives are achieved. The build cycle is an iterative phase within the release plan. With the number of releases and the number of build cycles having been defined in earlier phases, each build is responsible for reaching certain goals laid out in advance.

As shown in Figure 3-2, the core of the build cycle phase is the repeatable programming cycle. It is here that each use case is implemented within the WebSphere Commerce programming model such that it satisfies the requirement. The user interface is built and the data model is actualized in the data store.

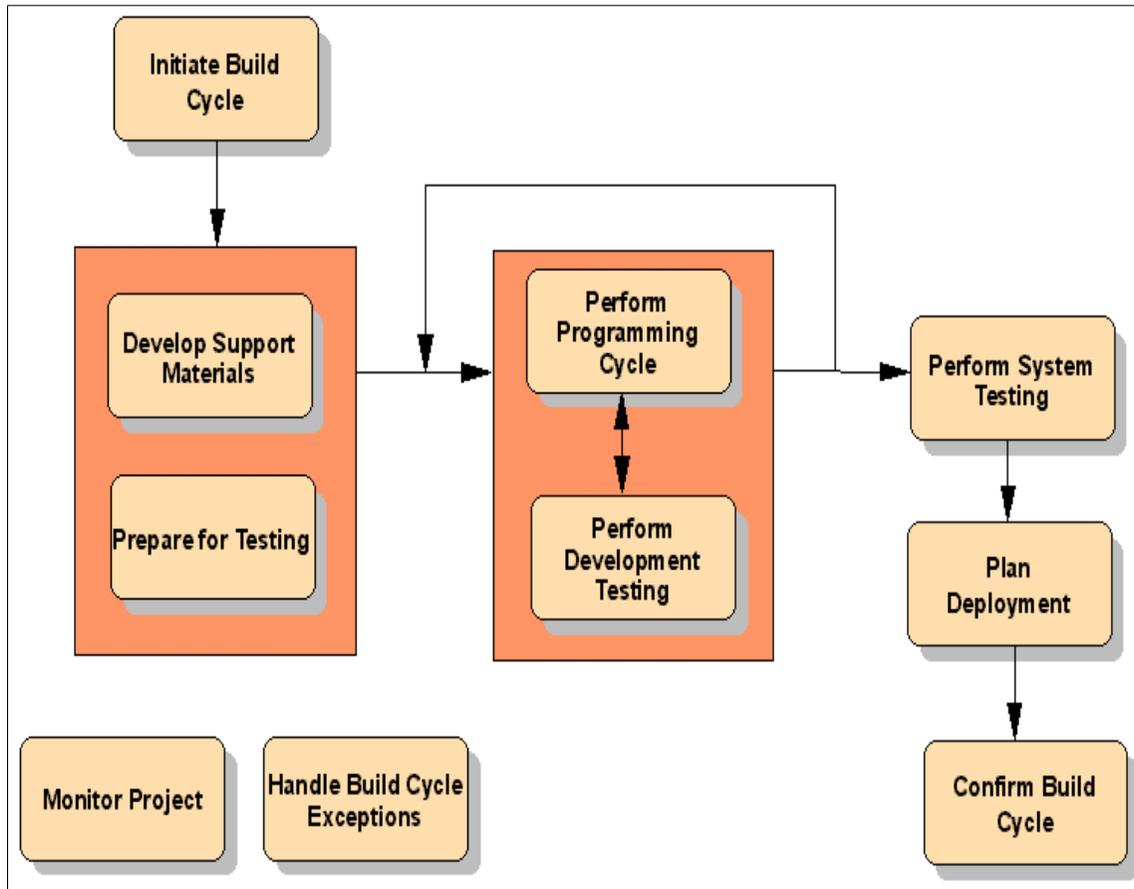


Figure 3-2 Build cycle process

Testing also forms a critical part of the build cycle phase. There are several phases of testing that occur at different points. Bound up with the programming cycle is a group of tests that can be performed by the development team: static testing and unit testing. This testing should occur as each programming cycle concludes. A separate testing team may also be engaged to perform a functional test on each component to ensure that the requirements for the component have been met. At the end of a series of programming cycles, when the build cycle has neared its conclusion, the test team should perform a system test wherein they integrate the various pieces of the application in order to test interoperability, performance and usability. Detailed information on testing a commerce site can be found in Chapter 6, “Testing a store” on page 169.

The nested iterative approach of the build cycle and the programming and testing cycles within it allows for a high degree of focus and specialization for each component of the commerce site. The final task in the build cycle phase is to plan for deployment. The lessons learned during the build cycle play a large role here as the plans originally developed in earlier phases are refined and the solution takes the final step to deployment.

3.1.6 Deployment

As its name implies, the primary purpose of the deployment phase is to launch the release of the system. A secondary goal is to prepare for the next release.

A round of user acceptance testing may be performed here while simultaneously preparing the production environment. The production environment will conform to the architectural standards established in the solution outline phase and refined throughout the process.

Once all of the approvals have been obtained, the cutover to production can occur. Following the deployment of the commerce site, a review of the release should occur in order to start planning for follow-on releases.

3.1.7 Solution Close

The solution close phase is a formal closure to the entire project. Solution startup and solution close are the bookends to every development effort and, just as solution startup is largely an introduction to the project, solution close is essentially a wrap-up. Activities include obtaining final customer signoff, capturing the project results and documenting all lessons learned during the engagement.

3.2 Example store requirements

The approach we take in this book is to display the customizable nature of WebSphere Commerce Business Edition by extending several of the sample stores provided with the product: InFashion, WebFashion and ToolTech. Each of our examples, detailed in **<INSERT X-REF TO PART 2>**, starts with one of these basic stores and alters it based on a set of requirements unique to each example. In order to provide you with some actual examples of requirements gathering as outlined above, a sampling of this activity is included here.

Through the creation of use cases, non-functional requirements and sequence diagrams, we begin to define the application model for some of our extended stores. The requirements we describe here are based on the sample store use cases that can be found in the online help files at:

http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

3.2.1 General store requirements

There are, of course, many decisions to make that apply to almost all stores. These decisions will lead to some general store requirements and should be discussed and settled upon at the outset of the requirements gathering phase. Examples of these basic considerations are outlined below.

- ▶ Store scope
 - a. Business type: business-to-consumer or business-to-business
 - Business-to-consumer stores will require straightforward functionality, an obvious and shopper-friendly shopping flow and, potentially, personalization.
 - Business-to-business stores will lead to requirements that include an interface that is very quick, tailored product catalogs and interfaces with systems outside of your company.
 - Refinements within these models may lead to more specialized requirements. For example, an auction model will require a bidding flow that is not needed in other business-to-consumer models.
 - b. Globalization and localization
 - Language support between and within countries
 - Currency requirements
 - Product definitions in light of local laws
 - Taxation
 - Look-and-feel customization
- ▶ Customers
 - a. Registered users
 - Regular shopper
 - Personalization requirements
 - Discounts based on usage
 - b. Non-registered users
 - Guest shopper usage
 - Retention of one-time only shopper data
- ▶ Products

- a. Definition and categorization
- b. Pricing
 - Sales
 - Method of discounting
 - Display
- ▶ Payment
 - a. Forms of payment accepted
 - Credit card processing method
 - Checking authorization
 - b. International payments
 - c. Shipping requirements

Following the definition of the site's requirements, you can create a store flow diagram to lead you through the development of the site. Use cases are particularly useful here as you can utilize them to ensure completeness of the flow diagram. Figure shows a simple flow diagram for the WebSphere Commerce out-of-the-box InFashion sample store as shown in the online help.

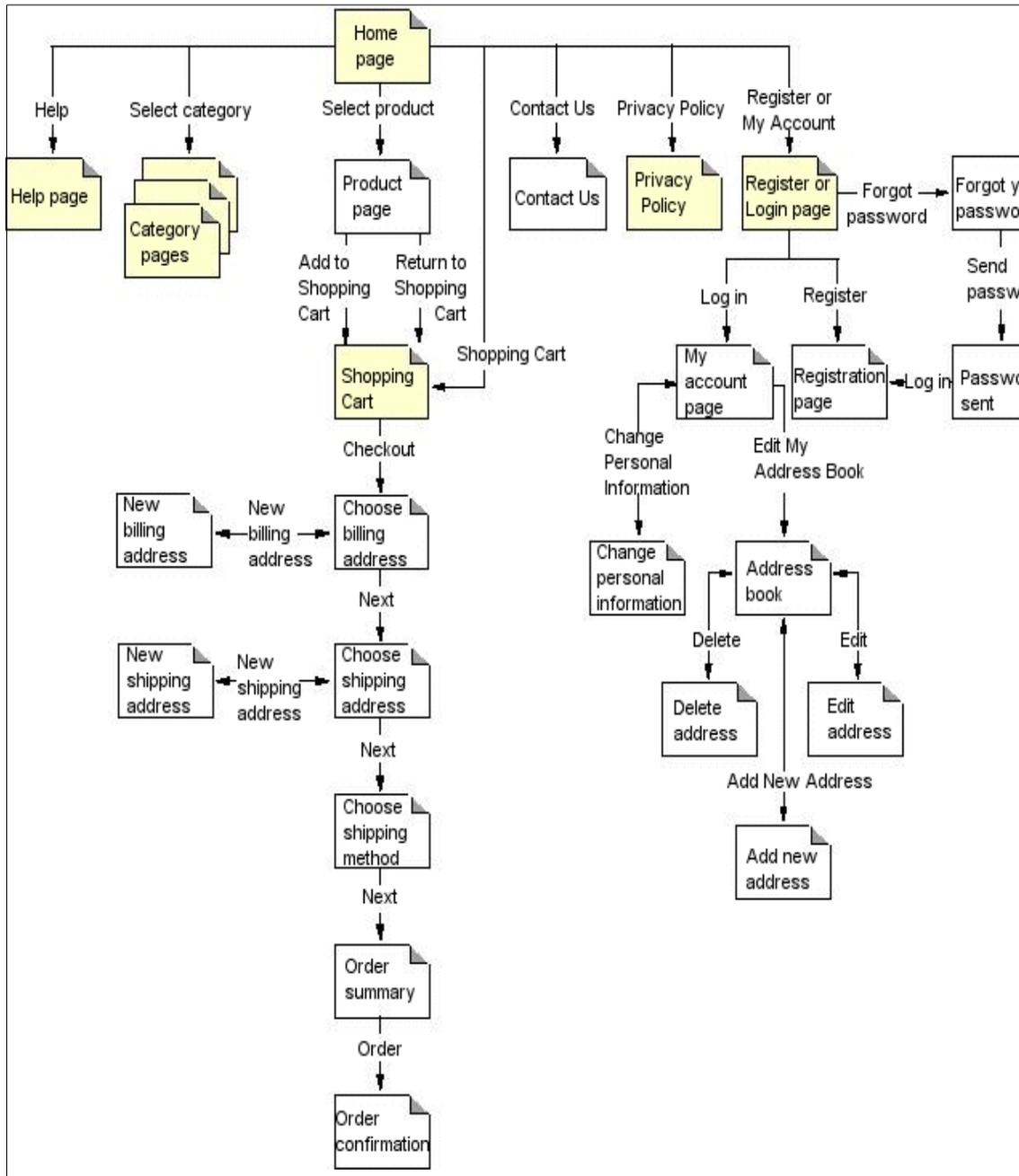


Figure 3-3 InFashion store flow diagram from WebSphere Commerce online help

3.2.2 Quick order implementation

The example which displays the quick order function, detailed in Chapter 9, “Orders” on page 233, is an extension of the ToolTech store. It allows a shopper to process orders that they often repeat without all of the associated processes that normally accompany a shopping experience. This is particularly useful in a business-to-business setting.

Functional requirements

The use case below defines the usage of the quick order function.

- ▶ Actor
 - Customer
- ▶ Preconditions
 - The customer has a valid account with the store.
 - The customer knows the product information (the product SKU, for example) for the products to be ordered.
- ▶ Main flow
 - a. A customer successfully logs in to the store.
 - b. The home page is returned to the customer, who clicks on **Enter More Items** under the Quick Order heading.
 - c. The system returns the quick order page. The customer enters the product number information and amounts for all of the products he or she wishes to order.
 - d. The customer clicks **Order**.
 - e. If one or more of the product numbers cannot be found, the Exception Flow occurs. Otherwise, the shopping cart is displayed with the customer’s order information in it
- ▶ Exception flow
 - a. If one or more of the items the customer has entered in the quick order page cannot be found, then a confirmation page will be returned. The confirmation page lists the order and informs the customer of which products could not be found.
 - b. The customer edits the items in place, correcting any mistakes or clicks **More Items** which returns the use case to the Main Flow.
- ▶ Postconditions
 - The customer has a shopping cart with the items ordered from the quick order process.

System sequence diagram

A system sequence diagram shows a high-level flow of information and transactions between two entities. It is a helpful way to show the necessary interactions between users and systems as it displays the information in an easily readable form. A system sequence diagram can show multiple use cases or a single use case. Figure 3-4 shows a simplified system sequence diagram for the quick order function. The flow follows the customer from login to placing the order in the shopping cart.

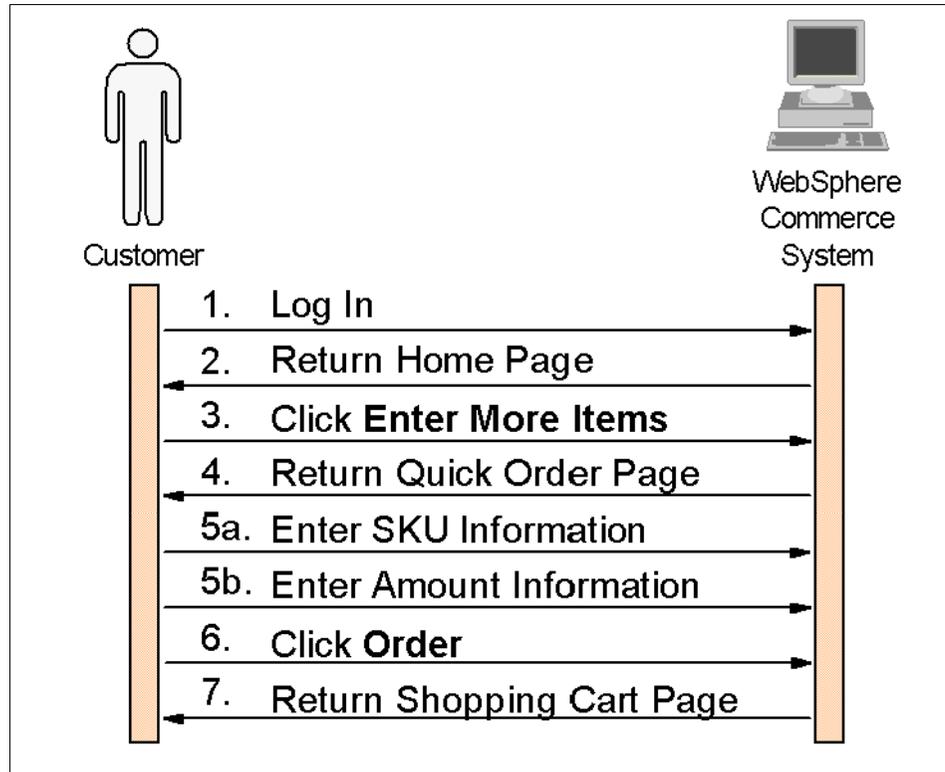


Figure 3-4 Quick order system sequence diagram

3.2.3 Price display including tax

In Chapter 10, “Shipping and taxes” on page 259, we show how to implement the display of prices where tax is included.

Functional requirements

There are several use cases involved with altering the displayed price to include taxes. As an example, one of these use cases is shown below. It addresses the store administrator's creation of a new tax category wherein taxes are included in the displayed price.

- ▶ Actor
 - Store administrator
- ▶ Preconditions
 - A store has been set up and configured.
- ▶ Main flow
 - a. A store administrator logs into WebSphere Commerce Store Services, selects the store to be operated upon and clicks **Tax**.
 - b. The administrator clicks **Categories** from the left side navigation bar. The system displays the Tax categories page for the store.
 - c. The administrator types in the name of the new tax category he or she wants to create, selects **Sales tax**, clicks the **Include tax in display price** checkbox and click **Add**.
 - d. The system creates the new tax category which may be applied to a section of the product catalog.
- ▶ Postconditions
 - A tax category which includes the tax in the price of the product.

3.2.4 Product creation via MQ

This example, implemented in Chapter 11, “Messaging customization” on page 291, examines the design of a method to create a product on a back-end system and import that new product to the WebSphere Commerce system via MQSeries.

Functional requirements

The use case below addresses the action of the store administrator creating a new product.

- ▶ Actor
 - Store Administrator
- ▶ Preconditions
 - A store has been set up and configured.
- ▶ Main flow

- a. A store administrator creates a product on a back-end system. The product is associated with a category and has a price and any required attributes.
 - b. The back-end system will create an XML message describing the product and place it on an MQSeries queue.
 - c. An MQSeries adapter in the WebSphere Commerce system will receive the message from the queue, parse it and call MQProductCreate.
 - d. MQProductCreate will create the product in the commerce database. If the product already exists, then the Alternate Flow occurs.
- ▶ Alternate flow
 - If the product described in the XML message already exists in the system, then the MQProductCreate command will update the attributes of the product in the database with the information from the XML message.
 - ▶ Postconditions
 - A new product exists in the store.

Non-functional requirements

Non-functional requirements describe how the system will accomplish the tasks defined in the functional requirements. They typically address items such as: performance, reliability, extensibility, development standards and system requirements. Some of the non-functional requirements for the above requirement include:

- ▶ An inbound-only MQSeries channel will be set up on the WebSphere Commerce system.
- ▶ A product will exist in the commerce system within 5 minutes of having been created on the back-end system.
- ▶ The commerce system will not receive any confidential information regarding the company in the creation of a product.

3.2.5 Welcome page based on role

This example uses the concept of roles in WebSphere Commerce to display a tailored page for those users who are assigned specific roles. If a user has the Buyer Approver or Buyer Administrator role, then he or she will be served a welcome page with links to tasks that are specifically designed for them: RFQ creation and order approval.

Functional requirements

The use case for the altered login flow is shown below.

- ▶ Actor
 - Customer
- ▶ Preconditions
 - The customer has previously registered with the store.
- ▶ Main flow
 - a. The system displays the logon page.
 - b. The customer selects a language and enters the userid and password in the appropriate fields. If the customer has forgotten the password, he or she can click **Forgot Your Password?** leading to the Alternate Flow.
 - c. The customer clicks **Submit**. The system checks the authorization information. If the user has entered an incorrect userid or password, the Exception Flow occurs.
 - d. The customer is transferred to the home page of the store. If the customer's roles include Buyer Approver or Buyer Administrator, the customer will be presented with two extra links: one for RFQ creation and one for order approval. Otherwise, these links do not appear.
- ▶ Alternate flow
 - a. The customer clicks on **Forgot Your Password?**
 - b. The system displays a screen where the customer enters the userid and clicks **Send My Password**.
 - c. The system sends an e-mail containing the password for the account to the e-mail address associated with the account.
- ▶ Exception flow
 - a. The customer enters an incorrect userid or password.
 - b. The system displays a screen informing the user of the error and the use case starts from the beginning.
- ▶ Postconditions
 - The customer is properly logged in to the system.

3.2.6 Amount-based order approval

In the use cases describing this example, a buyer will attempt to make a purchase that is over an amount that has been previously set against the contract. When that limit is met or exceeded, the purchase must be approved by a second party.

Functional Requirements

The first use case shown refers to the buyer.

- ▶ Actor
 - Buyer
- ▶ Preconditions
 - The buyer has finished shopping and has a contract number to use for the purchase.
 - The contract to be used has been set to require approval from another source when the purchase amount is in excess of \$250.
- ▶ Main flow
 - a. The buyer clicks **Check out**. The system returns a display of contract numbers which the buyer is entitled to use.
 - b. The buyer selects a contract from the list and clicks **Add to Order**. If the order is under \$250, then the system processes the order.
 - c. If the system is over \$250, the system puts the order into pending approval state and the buyer is informed that the order is awaiting approval.
- ▶ Alternate flow
 - The buyer's purchase exists in the system in the pending state.

The second example use case for this sample store, shown below, uses the postcondition from the first use case as input and addresses the approver's action.

- ▶ Actor
 - Approver
- ▶ Preconditions
 - A buyer in the approver's organization has attempted to make a purchase over \$250.
 - The purchase exists in the system in the pending state.
- ▶ Main flow
 - a. The approver clicks on **Order Approval** to get a list of orders in the pending approval state.
 - b. The approver reviews the order that the buyer has assembled and clicks on **Approve** which causes the system to process the order.
- ▶ Postconditions
 - The buyer's order has been processed.

Use case diagram

From a use case description, we can create a diagram, a graphic representation of the use case. The use case diagram shown in Figure 3-5 is for the approve order in amount-based approval system use case.

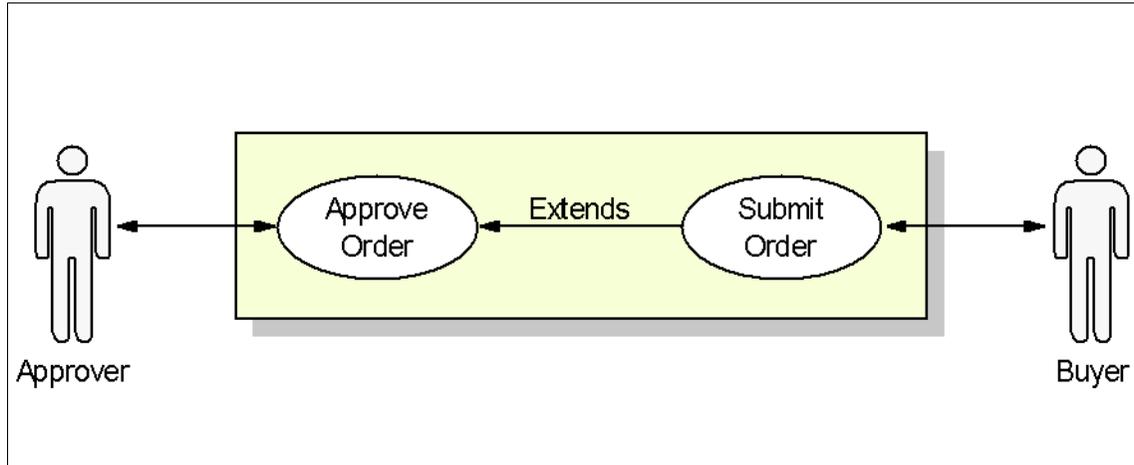


Figure 3-5 Approve order in amount-based approval system use case diagram

3.2.7 Contract-based logon

In the out-of-the-box ToolTech sample provided with WebSphere Commerce Business Edition Version 5.4, the flow is set up in such a way that the buyer selects a contract to use as a payment method after he or she has finished shopping. This example examines a scenario where the customer chooses a contract at the beginning of the shopping experience and is shown only the products and prices associated with that contract.

Functional requirements

The following use case addresses the requirement that a buyer choose a contract at the beginning of a shopping experience.

- ▶ Actor
 - Customer
- ▶ Precondition
 - A customer who is registered with the store accesses the site using a supported browser.
- ▶ Main flow
 - a. The customer accesses the store's login page.

- b. The customer selects a language from the drop down menu and enters the userid and password in the respective fields. If the customer has forgotten the password, he or she can click **Forgot Your Password?** leading to the Alternate Flow.
 - c. The customer clicks **Submit**. The system checks the authorization information. If the user has entered an incorrect userid or password, the Exception Flow occurs.
 - d. The system returns the home page of the store to the customer. From this page the customer selects the contract to be used throughout the shopping experience.
- ▶ Alternate flow
 - a. The customer clicks on **Forgot Your Password?**
 - b. The system displays a screen where the customer enters the userid and clicks **Send My Password**.
 - c. The system sends an e-mail containing the password for the account to the e-mail address associated with the account.
 - ▶ Exception flow
 - a. The customer enters an incorrect userid or password.
 - b. The system displays a screen informing the user of the error and the use case starts from the beginning.
 - ▶ Postconditions
 - A contract number is associated with the customer for the duration of the shopping experience.

3.2.8 Product bundles

Store administrators can make the shopping experience faster and easier by implementing bundles. A bundle is a collection of related catalog entries. When a shopper adds a bundle to the shopping cart, the bundle is decomposed into its' individual components. These components may individually be modified or removed from the cart by the shopper. A powerful marketing tool, bundles are helpful to the shopper because they make the shopper aware of available associated products.

Functional requirements

In Chapter 10, “Shipping and taxes” on page 259, we look at the creation of bundles. The following sample use case shows a user’s interaction with a bundle in a store.

- ▶ Actor

- Customer
- ▶ Preconditions
 - A bundle has been defined in the store.
- ▶ Main flow
 - a. The customer selects a bundle from the category page.
 - b. The system retrieves the bundle information and displays a page with detailed information about each product in the bundle. The bundle page includes the following information:
 - Product name
 - Brief product description
 - Product price
 - Product image
 - Appropriate product attributes
 - c. Where necessary, the customer selects the desired attributes for each product in the bundle. The customer then clicks **Add to shopping cart**.
 - d. The system displays the shopping cart with the individual components listed. The customer can modify or remove the components as desired.
- ▶ Postconditions
 - The customer has the desired items in the shopping cart.



Planning and development

Setting up the development environment properly and efficiently together with having an organized view of the structure of the development environment is a good start for an e-commerce project. This chapter provides guidance for the process of setting up the development environment, but also provides a short description to the products necessary as well as some planning needed prior to the set-up of the development environment.

The chapter is organized into the following sections:

- ▶ Development overview
- ▶ Planning WebSphere Commerce development
- ▶ Install the development environment
- ▶ Post-install configuration
- ▶ WebSphere Studio Application Developer

4.1 Development overview

Developing e-commerce solutions requires a variety of tools and there are many tools for the developer to choose from. IBM provides WebSphere Commerce Studio to help the developers achieve the expected results and this can be used with WebSphere Commerce Business Edition as well with WebSphere Commerce Professional Edition.

WebSphere Commerce Studio provides the main tools necessary to build a WebSphere Commerce solution. WebSphere Commerce Studio is available in a Business Developer Edition and a Professional Edition for Windows NT or Windows 2000. The two editions contain the same products apart from the edition of WebSphere Commerce chosen.

This section will provide a brief introduction to WebSphere Commerce Studio and the components which it contains.

As an addition to the products that are included in WebSphere Commerce Studio we do a brief introduction to WebSphere Studio Application Developer in 4.5, “WebSphere Studio Application Developer” on page 96.

Important: WebSphere Commerce Studio is the officially supported development environment for use WebSphere Commerce Business Edition V5.4. At the time we wrote this redbook, WebSphere Studio Application Developer V4 was not supported for development with WebSphere Commerce Business Edition V5.4. We include instructions for configuring WebSphere Studio Application Developer to work with WebSphere Commerce Business Edition, but this integration has not been fully tested and is not officially supported. Please remember that our instructions are for your guidance only and we cannot guarantee that they will work in your environment.

4.1.1 WebSphere Commerce Studio V5.4

The WebSphere Commerce Studio package provides you with the tools required to develop front-end as well as back-end assets for your e-commerce application. It is only available as a package for Windows NT and Windows 2000, but the output can be published to any WebSphere Commerce server of your choice.

The products that are included in the WebSphere Commerce Studio V5.4 package are:

- ▶ WebSphere Studio Application Developer

- Page Detailer V3.5
 - Page Designer
 - Applet Designer
 - IBM Distributed Debugger
 - Commerce Studio extensions
- ▶ VisualAge for Java, Enterprise Edition V4.0
 - ▶ Blaze Advisor Builder V3.1.3
 - ▶ Blaze Innovator Workbench V3.1.3
 - ▶ XML Tools
 - ▶ WebSphere Commerce Developer Edition V5.4

WebSphere Studio V4.0, Advanced Edition

WebSphere Studio V4.0, Advanced Edition, contains of a collection of tools that are used and accessed through a common interface. WebSphere Studio enables collaborative work in an easy to use fashion that increases the productivity and decreases the start-up process of a teamwork environment. This section provides a short introduction to the components included in WebSphere Studio.

For more detailed information, refer to

<http://www.ibm.com/software/webservers/commerce/>

Page Detailer v3.5

The Page Detailer is a tool used for analyzing the content being sent from the Web server to the client browser. Its purpose is to make it easier for the developer and designer to isolate problem areas that might be causing long response times and to help validating that the information passed from the web server to the client are meeting the requirements set on the e-commerce solution in terms of total sizes of the pages presented on the client browser.

Page Detailer can help you improving the size of the content, the organization of the content and the delivery of the content through providing statistics of analyzed data being sent from the web server.

To get some tips on how to improve the performance of a web site, read the technical article *Improve Web Site Performance with WebSphere Studio Page Detailer* at

<http://www7.software.ibm.com/vad.nsf/data/document4361>

Page Designer

Page Designer is an advanced-function HTML editor integrated into WebSphere Studio that provides you with the ability to create and maintain HTML pages as well as JSP pages for your Web site. Page Designer includes two tools for working with graphics: WebArt Designer and AnimatedGif Designer.

Applet Designer

The Applet Designer is a tool that will help you to create simple Java applets. It uses visual composition to ease the process for the developer to combine different JavaBeans into applets.

IBM Distributed Debugger

To help debug your applications, WebSphere Studio includes the IBM Distributed Debugger. IBM Distributed Debugger offers the ability to debug Java objects when they are actually running on WebSphere Application Server.

The debugger provides a graphical user interface in which you can issue commands over the network to control the execution of the program running on WebSphere Application Server on a different machine. You can set breakpoints as in VisualAge for Java, step through your code and examine the value of variables in the code.

To find more information about the IBM Distributed Debugger, go to the WebSphere Developer Domain Library and search for IBM Distributed Debugger.

<http://www7b.software.ibm.com/wsdd/library/>

Commerce Studio extensions

The Commerce Studio edition of WebSphere Studio enhances the standard environment with two additional features that are WebSphere Commerce specific.

- ▶ You will have the ability to publish your store assets to the store archive file (SAR file) and publish them onto the WebSphere Commerce server in one step.
- ▶ You can import a store archive file and edit its contents in WebSphere Studio and when done, you can export the content back to the store archive file.

VisualAge for Java, Enterprise Edition V4.0

WebSphere Studio is used for creating and maintaining front-end assets such as HTML and JSP pages while VisualAge for Java, Enterprise Edition, is used to create new business-logic in the form of commands and Enterprise JavaBeans to use with your WebSphere Commerce solution.

VisualAge for Java includes tools for developing and debugging JSP templates. You can use WebSphere Studio to develop the JSP templates and integration with VisualAge for Java enables you to easily add content to the created JSP templates.

VisualAge for Java includes a complete test environment named WebSphere Test Environment, which enables you to run all the WebSphere Commerce functions together with any code you might develop without needing to deploy your code to a WebSphere Commerce Server.

At the VisualAge Developer Domain Library, you can find more information about the VisualAge for Java product.

<http://www7.software.ibm.com/vad.nsf/Data/Document2001?OpenDocument&SubMast=1>

Blaze Advisor Builder V3.1.3

Blaze Advisor Builder is the development component used to create personalization and business rules to tailor the content to be individualized. This enables you to do up selling, cross selling and so on. Blaze Advisor Rule Engine and Server then implement the rules created in a live environment.

Blaze Innovator Workbench V3.1.3

The Blaze Innovator Workbench is a tool to be used by developers to define editable rule service components. These components are then implemented into an administrative Web interface that will enable non-technical people to maintain the rules in an easy and logical fashion.

XML tools

The XML tools package contains tooling for XML. It allows for creating, transforming, and querying XML documents through the use of visual tools. The version of the tools provided in the package are not intended for use in a production environment, but are only provided as a preview of the final XML tools bundle.

It contains the following tools:

- ▶ Visual DTD
- ▶ Visual XML Creation
- ▶ Visual XML Builder
- ▶ Visual XML Transformation
- ▶ Visual XML Query

WebSphere Commerce Developer Edition V5.4

The WebSphere Commerce Studio package includes a version of WebSphere Commerce for Windows to be used for testing and development purposes only. If you plan to put the site into production, you need to need to purchase a licensed copy of WebSphere Commerce.

4.2 Planning WebSphere Commerce development

Prior to starting the development, a planning stage is necessary to get an efficient and secure team environment to work in. This section will mention some options available to you and your team. We discuss the following topics:

- ▶ Development initialization phase
- ▶ Team development

4.2.1 Development initialization phase

This phase of a development is very important and should not be taken lightly, as it sets the ground for the development foundation. The initialization phase should be used wisely to optimize the startup process of the project and to create processes that are to be used later in the development cycle, such as test, deployment, and packaging. Having a solid ground of routines, methods and standards will make the development more efficient and understood by all members in a team.

The team itself creates the processes and uses any standards set by the organization they are working for. In the end, it is the team that makes the decision so it is very important that the routines, methods and standards are decided early in the development phase and commonly known by all team members.

Things to consider in this phase include set coding standards, the way of setting up the development environment, which tools to use, how to deploy and test and so on.

Choosing your tools

WebSphere Commerce Studio V5.4 is only available on Windows NT and Windows 2000. However the outcome from the tools in WebSphere Commerce Studio is applicable to any server running WebSphere Commerce. The options for configuration of WebSphere Commerce Studio are a one-tier or a two-tier setu as follows:

- ▶ Install WebSphere Commerce Studio and the database on the same machine.

- ▶ Install WebSphere Commerce Studio on one machine and the database on a remote machine.

The advantages of using a remote database in the WebSphere Commerce Studio development environment are:

- ▶ As a team, you can share a database. This will minimize the database setup as well as the database maintenance. Moreover the increased level of integration this provides may mean that validation of the code the team produces will be minimized. The team members can utilize their own database for experimenting with set up and then use the remote database for real data testing.
- ▶ It will provide better availability of system resources on the local machine to other applications.

The disadvantage is that you are depending on access to the remote machine where the database resides.

WebSphere Commerce Studio provides the ability to function with an Oracle database or an IBM DB2 database. The development environment should use the same components as the production environment, so the choice of the development database should be based on the production requirement.

4.2.2 Team development

This section describes the options you have in choosing a system for sharing the development code amongst a team. Developing code in a team is not as simple working alone, as it involves a shared responsibility over the code that is being produced, as well as group to the access to the code that is produced. Therefore a good versioning control system, together with established routines for code sharing, are needed to make the code maintenance easier.

WebSphere Studio

WebSphere Studio Advanced Edition supports a number of source configuration management systems (SCM) which can be integrated with WebSphere Studio using Microsoft's Common Source Code Control (SCC) interface. This means that the integration to the SCM system is very easy to do. If any of the supported SCM systems clients is installed on the local machine, WebSphere Studio will automatically recognize the application and no further integration is necessary. WebSphere Studio currently supports Rational ClearCase, Merant PVCS, IBM VisualAge TeamConnection and Microsofts Visual SourceSafe.

VisualAge for Java

VisualAge for Java, Enterprise Edition, ships with a repository server (EMSRV) that manages concurrent access to a shared repository. Using EMSRV does not require any source configuration management (SCM) tool as the repository system is integrated into VisualAge for Java. As stated in the *IBM VisualAge for Java Online Help*, VisualAge for Java, Enterprise Edition, is different to other SCM systems in the following ways:

- ▶ Team developers do not reserve or check out program elements, so program elements are always available to everyone on the team.
- ▶ There is no need to check in a program element after changing it. Incremental changes are immediately saved in the shared repository.
- ▶ Anyone on the team can access and modify any program element for development, testing, and debugging purposes, regardless of who owns the program element. This facilitates code reuse and collaborative development.
- ▶ Change is managed at the object level rather than at the file level. This facilitates parallel development of classes by more than one developer.
- ▶ Program element owners approve changes by releasing them into the team baseline. There is an emphasis on roles and responsibilities assumed by the team, rather than on file locking performed by the software.

As an alternative to the EMSRV repository server, VisualAge for Java V4.0 contains a source configuration management bridge to provide enhanced integration to external SCM systems. It currently supports Rational ClearCase, Merant PVCS, IBM VisualAge TeamConnection and Microsofts Visual SourceSafe. The bridge works in the manner that when the of VisualAge for Java is out of sync with the external SCM, the developer will receive a visual reminder in VisualAge. Tooling is provided for updating the external SCM with the internal repository of VisualAge for Java. When developing, the developer is still utilizing the local repository of VisualAge for Java.

Depending on the size of the development team and the way the team is developing a WebSphere Commerce application, one of the above choices may be selected. The third alternative is to use local repositories only, that is every developer is responsible for their own repository and the team uses a SCM system without any integration with VisualAge for Java. In this case it is difficult to keep track of changes and to have a common backup of the code produced, so we do not recommend this set up.

Often existing conditions influence the decision as to which SCM solution is adopted. For example, if all other versioning of project related documentation resides in an external SCM it is more likely that the code generated within VisualAge for Java will be stored in the same system.

For more information on the VisualAge for Java version control support, refer to:

- ▶ The redbook *VisualAge for Java Enterprise Version 2 Team Support*, SG24-5245
- ▶ The *IBM VisualAge for Java Online Help*. Select **Topics -> Concepts -> Team development**.

Integration

WebSphere Studio V4.0 integrates closely with VisualAge for Java V4.0. It provides the developer with quick navigation between the two products in the following cases:

- ▶ When you update a file in one product, you can also update the copy in the other product.
- ▶ When you use the wizards in WebSphere Studio to develop servlets or JavaBeans, you can transfer the servlets or JavaBeans to VisualAge for Java to extend them and maintain them.
- ▶ If you use VisualAge for Java to develop servlets or JavaBeans, you can transfer them to WebSphere Studio to use in the JavaBean wizard or publish to your Web sites.

If you will be utilizing the integration possibilities of these products and are using a versioning control system, it is recommended that you will use the same versioning system with VisualAge for Java.

4.3 Install the development environment

This section guides you through the manual procedure of installing WebSphere Commerce Studio and Application Assembly Tool (AAT) that might be necessary if you are developing back-office business logic like EJBs.

The installation process is organized in the following sections:

- ▶ Pre-installation requirements
- ▶ Install VisualAge for Java, Enterprise Edition
- ▶ Install WebSphere Studio
- ▶ Install WebSphere Commerce Studio
- ▶ Install Application Assembly Tool

4.3.1 Pre-installation requirements

This section summarizes the minimum hardware and software requirements needed to set up a development environment for developing WebSphere Commerce V5.4 components.

The installation process for a complete setup of WebSphere Commerce Studio is quite time-consuming and hence it is important to get it right the first time. Therefore we recommend that you carefully follow the installation instructions and the first thing to review is that you will meet all pre-installation requirements.

Hardware

Ensure that you meet the following hardware requirements before you install WebSphere Commerce Studio:

- ▶ A minimum of 512 MB of random access memory (RAM).
- ▶ A dedicated IBM-compatible personal computer with a Pentium III 733 MHz processor.
- ▶ A local area network adapter that supports the TCP/IP protocol.
- ▶ Depending on which components of the WebSphere Commerce Studio you choose to install, you will need the following amount of disk space on your computer:
 - To develop store-front assets you will need a total of 335 MB. The specific components requires the following:
 - 300 MB for WebSphere Studio
 - 25 MB for Store Archive Tools
 - 10 MB for Blaze Advisor and Blaze Innovator Workbench.
 - To develop back-office business logic you will need a total of 2 GB. The specific components requires the following:
 - 1.2 GB for VisualAge for Java, Enterprise Edition
 - 500 MB for IBM DB2 Universal Database
 - 300 MB for WebSphere Commerce Development Environment
 - 160 MB for IBM Distributed Debugger
 - 15 MB for Applet Designer
 - 10 MB for Page Detailer

Furthermore you will need an additional 50 MB on your C: drive. If the partition is formatted with FAT partitioning and it is larger than 1 GB, you will need 100 MB available on your C: drive.

Software

It is very important that you have met the following software requirements before installing WebSphere Commerce Studio:

- ▶ Ensure that you have one of the following operating systems installed:
 - Windows 2000 Server Edition, or Advanced Server Edition with Service Pack 2 applied. You can find the required updates at the following URL:

<http://www.microsoft.com/windows2000/downloads/servicepacks/>

- Windows NT Server Version 4.0 with Service Pack 6a installed on your WebSphere Commerce Studio machine. You can find the required updates at the following URL:

<http://www.microsoft.com/ntserver/nts/downloads/>

- Microsoft Internet Explorer V5.5 with Service Pack 1 with the latest security patches on all the machines you will use to access WebSphere Commerce Studio. Internet Explorer can be downloaded at the following URL:

<http://www.microsoft.com/windows/ie/downloads/>

- If you plan to install VisualAge for Java, Enterprise Edition, please review the notes in “Considerations prior to installation” on page 82.

Important: Other versions/editions of the software requirements described above might work with WebSphere Commerce Studio but IBM does not support them.

Other requirements

You have to be logged on to the machine with a Windows user ID that has Administrator privileges to be able to install your applications. To achieve maximum control of user IDs appointed to be used with the applications, create a new dedicated administrator user to be used for this purpose.

Tip: Do not use an administrator userid that is not being validated locally, but at a domain, as this might cause problems when you have to change your password.

Pre-installation advice

If you are running any antivirus software, change its startup type to Manual in the Services window and then restart your machine prior to installing WebSphere Commerce Studio. This will prominently speed up the installation process. Do not forget to change back to Automatic startup type when the installation of WebSphere Commerce Studio is complete.

4.3.2 Install VisualAge for Java, Enterprise Edition

This section provides detailed information on how to install VisualAge for Java, Enterprise Edition, and how to verify that the installation has been successful. The installation of VisualAge for Java is organized into three sections:

- ▶ Considerations prior to installation

- ▶ Install VisualAge for Java
- ▶ Verify the installation

Attention: VisualAge for Java, Enterprise Edition, cannot be installed next to the installation of WebSphere Commerce Studio and function properly without some manual configuration.

Considerations prior to installation

Before you start the installation of VisualAge for Java, Enterprise Edition, take notice of the following:

- ▶ The total classpath in VisualAge for Java must be less than 437 characters. After the installation, VisualAge for Java has taken the length of the installation path. Hence use a short installation path. For example c:\WCDev\VAJ.
- ▶ You must have at least 20 MB free on your Windows system drive, and your environment variable TEMP or TMP must point to a valid temporary directory with at least 6 MB free.

Tip: Verify which directory is currently in use for the TEMP and/or TMP variable by opening a Command Prompt, enter SET and press Enter. All system variables and its values will be listed.

For further information refer to *Installation and Migration Guide* that comes with the VisualAge for Java, Enterprise Edition CD1.

Install VisualAge for Java

To install VisualAge for Java, Enterprise Edition, complete the following steps:

1. Insert the IBM VisualAge for Java V4.0, Enterprise Edition CD 1 into the CD-ROM drive.
2. Using Windows Explorer from the root of the CD, double-click **Setup** to start the install.
3. When the IBM VisualAge for Java Install window appears, click **Install Products -> Install VisualAge for Java**.
4. When the Choose Setup Language window appears, select your national language from the drop-down menu (English United States is the default) and click **OK**.
5. When the Welcome window appears, click **Next**.
6. When the Software License Agreement window appears, read the agreement and if you agree to the terms, then select **I accept the terms in the license agreement**, and then click **Next**.

7. When Setup Type window appears, select **Custom by Scenario**, and then click **Next**.
8. When prompted for the selection options, select the check boxes shown in Figure 4-1 and click **Next**.

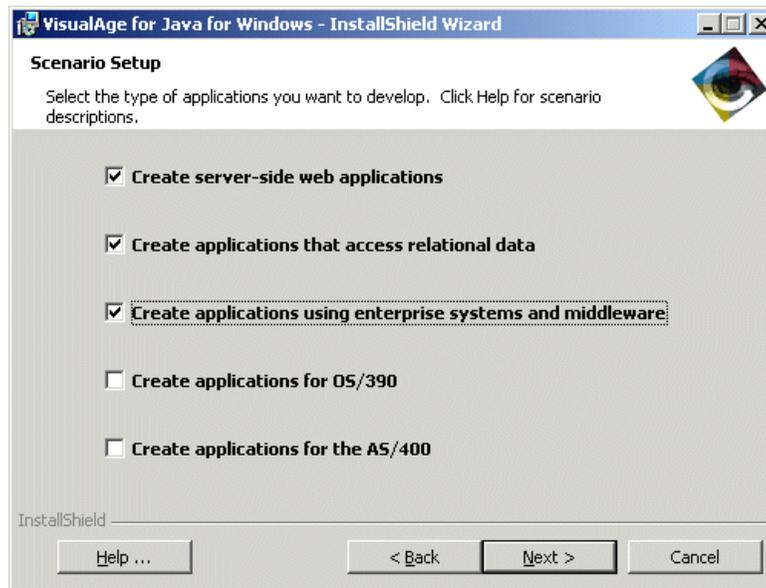


Figure 4-1 Scenario Setup window

9. When the Edit Features window appears, click **Change** to alter the install directory.
10. When the Change Current Destination Folder appears, enter the folder name: c:\WCDev\VAJ. Click **OK**.
11. Click **Next** to proceed.
12. When the Location of the Repository window appears, select **Local (default)**, and then click **Next**.
13. When the Ready to Install the Program window appears, click **Install**. You should see a window Installing VisualAge for Java for Windows with the status indicator.
14. When the InstallShield Wizard Completed window appears, click **Finish**.

Verify the installation

Verify that the installation of VisualAge for Java was successful by completing the following steps:

1. Start VisualAge for Java by clicking **Start -> Programs -> IBM VisualAge for Java for Windows V4.0 -> IBM VisualAge for Java**.
2. The first time you start VisualAge for Java, you will be prompted for a network name for the user called Administrator. Type your Windows user name as network name.

Attention: The prompt window may appear behind already opened other applications. If that would occur, minimize all the other applications by using the minimize button on the upper-right corner on each window. Do not use the minimize button on the quick-launch bar (Windows 2000).

3. If the workspace opens, the installation of VisualAge for Java, Enterprise Edition, has successfully been installed.
4. Close VisualAge for Java.

Tip: In VisualAge for Java, Enterprise Edition, the current workspace owner is shown on the title bar of the workspace.

4.3.3 Install WebSphere Studio

This section will guide you through a separate installation of WebSphere Studio and is organized into the following sub-sections:

- ▶ Install WebSphere Studio
- ▶ Verify the installation

Attention: WebSphere Studio cannot be installed next to the installation of WebSphere Commerce Studio and function properly without some manual configuration.

Install WebSphere Studio

1. Insert the WebSphere Commerce Studio, Version 5.4 CD into your CD-ROM drive.
2. Using Windows Explorer, locate the WebSphereStudio4 directory and double-click the **Setup** file to let the installation process commence.
3. When the Welcome window appears, read the instructions and when ready, click **Next**.
4. When the Software License Agreement window appears, read the entire License agreement and if you accept all terms of the License Agreement click **Yes** to continue the installation process.

5. When the Choose Destination Location window appears, you have the option to change the destination folder of your installation. To keep all the WebSphere Commerce development tools located under the same directory, click **Browse**.

When the Choose Folder window appears enter c:\WCDev\Studio40 in the Path text-field and click **OK**.

6. When the Setup window appears with a message The folder does not exist, would you like to create it?, click **Yes**.
7. Click **Next** to proceed.
8. When the Select Components window appears, ensure that the IBM WebSphere Studio V4.0 component is checked and click **Next**.
9. When the Select Program Folder window appears, accept the default folder (IBM WebSphere), and click **Next**.

Tip: If you would like to group the development tools together in the start-menu, add a backslash and the name of the sub-category of the new folder. For example IBM WebSphere\Development Tools.

10. When the Start Copying Files window appears, review the current settings of the installation and if you find them to be correct click **Next**.
11. When the Setup Complete window appears, select **Yes, I want to restart my computer**, and then click **Finish**.

Verify the installation

To verify the installation of WebSphere Studio by completing the following steps:

1. Start the application by clicking **Start -> Programs -> IBM WebSphere -> Studio 4.0 -> IBM WebSphere Studio V4.0**.
2. When the Welcome to IBM WebSphere Studio window appears, select **Create a new project using:** and from the drop-down menu select **Default Template**. Click **OK**.
3. When the New Project window appears, enter a bogus project name in the Project Name text-field. For example TestProject. Click **OK**.
4. If the main workarea appears, WebSphere Studio has successfully been installed. Close WebSphere Studio.

4.3.4 Install WebSphere Commerce Studio

This section will provide a step-by-step installation guide of WebSphere Commerce Studio.

The installation of WebSphere Commerce Studio is organized into the following sections:

- ▶ Installation requirements
- ▶ Install WebSphere Commerce Studio
- ▶ The next step

Installation requirements

Prior to installing WebSphere Commerce Studio, please ensure that the following requirements are met:

- ▶ You are using a Windows userid that has Administrator privileges and the name of the userid does not exceed 8 characters.
- ▶ If you plan to use an Oracle database, the database need to be installed prior to installing WebSphere Commerce Studio. Please refer to Chapter 3, Installing an Oracle Database, in *IBM WebSphere Commerce Studio for Windows NT and Windows 2000 Installation Guide Version 5.4*.
- ▶ Ensure that the hardware and software requirements described in Chapter 4.3.1, “Pre-installation requirements” on page 79 are met.

Install WebSphere Commerce Studio

To install WebSphere Commerce Studio, complete the following steps:

1. Insert the WebSphere Commerce Studio CD into your CD-ROM drive.
2. With the use of Windows Explorer go to the root directory of the CD and double-click the **Setup** file to commence the installation.
3. When the Choose Setup Language window appears, click **Next**.
4. When the Welcome window appears, read the instructions and when done click **Next** to continue.
5. When the License Agreement window appears, read the entire License agreement and if you accept all terms of the License Agreement click **Accept** to continue the installation process. If you click Decline, the installation program will exit.
6. When the Select Components window appears, select the following:
 - **Develop Store Front Assets using WebSphere Studio.**
 - **Develop Store Back-Office Logic using VisualAge for Java**

In the Select Database drop-down list you select your choice of database to be used in the development environment. Click **Next** to continue.

Note: If you want to use Oracle Database as your WebSphere Commerce Studio database, you need to install the database prior to installing WebSphere Commerce Studio.

7. When the WebSphere Commerce Instance Information window appears, leave the default text in the text boxes as in Figure 4-2 and check **Include Sample Store**. Click **Next** to continue.

Note: To be able to perform the tutorials in the *IBM WebSphere Commerce Programmer's Guide*, the sample store is necessary to be selected in the WebSphere Commerce Studio installation.



Figure 4-2 WebSphere Commerce Instance Information window

8. When the Database Information window appears, enter the Windows userid used when installing the database in **DB Admin User** and **DB User**. If DB2 not yet reside on the system, enter the currently used administrator userid. Provide the corresponding password for the user in **DB Admin Password** and **DB User Password**. Click **Next** to continue.
9. When the Choose Destination window appears, you have the option to override the default installation path for each product that is to be installed. To keep all the development installations in the same parent directory, we

change the destination folders to c:\WCDev\CommerceStudio and c:\WCDev\CommerceServerDev. When done, click **Next**.

10. When the Select Program Folder window appears, accept the default folder (IBM WebSphere Commerce Studio), and click **Next**.
11. When the Summary window appears, review the installation configuration carefully and then click **Next**.
12. Depending on the software packages already installed on your system, you are prompted to insert the appropriate CDs for WebSphere Commerce Studio, DB2, VisualAge for Java, and WebSphere Commerce. Follow the on-screen instructions.
13. If you already have WebSphere Studio installed, a dialog as shown in Figure 4-3 will appear. Click **Yes** to add some registry settings to WebSphere Studio.

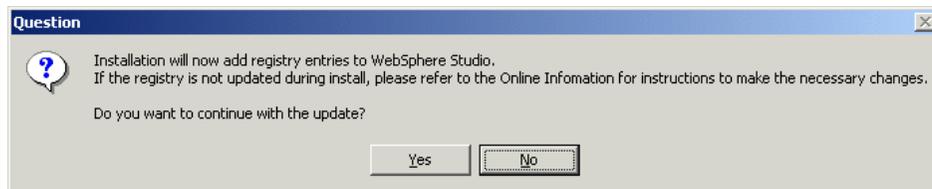


Figure 4-3 Add registries to WebSphere Studio prompt

14. When the Setup Complete window appears, select **Yes, I want to restart my computer now** and click **Finish** to restart your computer.

The next step

To finalize the installation of the WebSphere Commerce Studio refer to 4.4, “Post-install configuration” on page 90.

4.3.5 Install Application Assembly Tool

The Application Assembly Tool (AAT) is a component that ships with the IBM WebSphere Application Server V4.0. If you plan to develop back-office business logic you will need to install the Application Assembly Tool to be able to deploy your created Enterprise JavaBeans (EJBs) to your Enterprise Application.

This installation-guide assumes that WebSphere Application Server does not reside on the machine prior to the installation.

Install AAT

Perform the following procedure to install the Application Assembly Tool:

1. Insert the IBM WebSphere Application Server V4.0 Advanced Edition CD into the CD-ROM drive.
2. With the use of Windows Explorer, go to the root of the CD directory structure and double-click on the **Setup** file.
3. When the Choose Language Setup window appears, select **English** and click **OK**.
4. When the WebSphere Application Server 4.0 window appears, read the instructions carefully and then click **Next** to continue.
5. When the Installation Options window appears, select **Custom Installation** and click **Next**.
6. When the Choose Application Server Components window appears, de-select all components but **Application and Development Tools** and **IBM JDK 1.3.0**. Click **Next** to continue.

Note: If a JDK already is installed on the system, you can use that JDK by clicking **Other JDK** and in the window that popup specify the location. The installation program will alert you if the custom JDK does not meet the minimum requirements.

7. When the Product Directory window appears, you may modify the install path. To accept the default, click **Next**.
8. When the Select Program Folder appears, click **Next** to accept the default folders (IBM WebSphere\Application Server V4.0 AE).
9. When the Install Options Selected window appears, carefully review your selections and if they are ok click **Next** to continue.
10. When the Setup Complete window appears, click **Finish** to complete the installation.

Verify the installation of AAT

Verify the installation of AAT by completing the following steps:

1. Start AAT by clicking **Start -> IBM WebSphere -> Application Server V4.0 AE -> Application Assembly Tool**.
2. If the Welcome to Application Assembly Window Tool appears, the installation was successful.

4.4 Post-install configuration

This section will guide you through some post-configuration that is necessary before starting to develop WebSphere Commerce components. You must have installed WebSphere Commerce Studio prior to the post-configuration, refer to 4.3, “Install the development environment” on page 79.

To complete the post-install configuration, complete the instructions in the following sections:

- ▶ Configure VisualAge for Java, Enterprise Edition
- ▶ Create a sample store in VisualAge for Java
- ▶ Maintenance

4.4.1 Configure VisualAge for Java, Enterprise Edition

This section will provide a checklist to the post-configuration to the installation of VisualAge for Java, Enterprise Edition V4.0. The configuration of VisualAge for Java is quite extensive and is very good described in Chapter 10, “Configuring VisualAge for Java” in *IBM WebSphere Commerce Studio for Windows NT and Windows 2000*. Thus will this section only provide a checklist and some assistance to the instructions.

Import required files into VisualAge for Java

1. Apply the EJB-1.1-DeployedTool.zip fixpack.
2. Add features to VisualAge for Java.
3. Apply EJB-deletion-of-attributes e-fix. When done an asterisk will show next to readonly - Prevent invalid deletion of EJB read-only attributes, as in Figure 4-4.

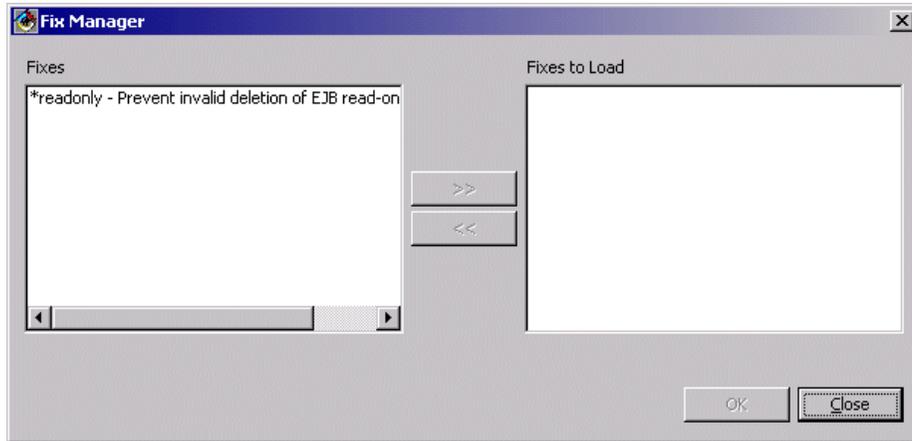


Figure 4-4 Fix Manager

4. Restart VisualAge for Java.
5. Ensure that the Administrator is the workspace owner.

Tip: In VisualAge for Java, Enterprise Edition, the current workspace owner is shown on the title bar of the workspace.

6. Create open edition of the IBM WebSphere Test Environment (WTE) project.
7. Apply IBM WebSphere Test Environment fix, PQ50519.jar.
8. Create open edition of IBM EJB Tools project.
9. Apply IBM EJB Tools fix, ivjfix35.jar. When you are versioning the IBM EJB Tools project you will be prompted for a version name. Select the default and click **OK**.
10. Import the WebSphere Commerce repository.
11. Change workspace owner to WCS Developer.
12. Save the workspace.

Configuring the EJB and PNS servers

1. Click the EJB tab in VisualAge for Java
2. Select all EJB groups whose names that start with **WCS**. And add them to the Server Configuration.

Tip: If you know that you will not use some of the EJBs, it will shorten the startup process if you do not add these to the server configuration.

3. Modify EJB server properties. It is very important that the Data Source name is spelled correctly. If you are not sure, copy the name from the instance XML file. If you followed the installation process described in this book, the XML file is to be found at the following location:
c:\WCDev\CommerceServerDev\instances\VAJ_Demo\xml\VAJ_Demo.xml
4. Open the WebSphere Test Environment Control Center.
5. Modify PNS server settings and click **Apply**.

Attention: WebSphere Application Server default setup is to use the same bootstrap port as the WebSphere Test Environment. Make sure that you do not have WebSphere Application Server running simultaneously as you will start WebSphere Test Environment as a conflict will occur.

6. Start the PNS server
7. Wait until the Console window outputs Server open for business.
8. Create Datasource Configuration. See Figure 4-5 for reference. The datasource will be added to the WebSphere Test Environment Control Center Datasource list.

Field	Value
DataSource name	Sphere Commerce DB2 DataSource VAJ_Demo
Database driver	COM.ibm.db2.jdbc.app.DB2Driver
Database URL	jdbc:db2:VAJ_Demo
Database type	JDBC
Description	
Minimum connections	1
Maximum connections	30
Connection timeout	300
Idle timeout	1800
Orphan timeout	1800

Figure 4-5 Add DataSource

- Restart the PNS server.

Tip: Now all the parameters have been set for the WebSphere Test Environment. Save the workspace to store the settings before proceeding.

Start the EJB server

- Start the EJB server
- Wait until the Console window outputs `Server open for business` for the EJB Server thread.

Note: This might take 25 - 35 minutes depending on your system specifications.

Configuring rules services

These steps are optional and only have to be completed if your WebSphere Commerce instance is using rules services. However, it might be a good idea to complete these steps now as you might come to implement rules services in the future.

- Open the WebSphere Commerce Development environment XML file. If you followed the installation process described in this book, the XML file is to be found at the following location:
c:\WCDev\CommerceServerDev\instances\VAJ_Demo\xml\VAJ_Demo.xml
- Ensure that the RulesServices has all boolean variables set to **false**.
- Create a new project named **Rules Resources**.
- Import resources (not classes) to the Rules Resources project from the following JAR files:
 - c:\WCDev\CommerceServerDev\blaze\AdvSrv31\lib\AdvCommon.jar
 - c:\WCDev\CommerceServerDev\blaze\AdvSrv31\lib\Advisor.jar
 - c:\WCDev\CommerceServerDev\blaze\AdvSrv31\lib\AdvisorSrv.jar
 - c:\WCDev\CommerceServerDev\blaze\AdvIrt31\lib\InnovatorRT.jar

Tip: The workspace has now been modified and we recommend that you save the workspace before proceeding.

Starting the servlet engine

- Edit the classpath of the servlet engine.
- Start the servlet engine.

3. Verify in the Console window that the servlet has started. The servlet thread will output quite a lot of information and it will end with message likethose in Example 4-1. If it does not look similar to Example 4-1, it will not work. Then stop the servers in the following order: Servlet Engine, EJB Server and PNS server, and restart them in then start them in opposite order as described previously.

Example 4-1 Servlet Engine is started

```
Hostname bindings:  
[hostname-binding]: hostname=localhost:8080----> servlethost=default_host  
[hostname-binding]: hostname=127.0.0.1:8080----> servlethost=default_host  
[hostname-binding]: hostname={Machine name}:8080----> servlethost=default_host  
[hostname-binding]: hostname={Machine IP}:8080----> servlethost=default_host  
  
***Servlet Engine is started***
```

Setup store

Chapter 10, “Configuring VisualAge for Java” in *IBM WebSphere Commerce Studio for Windows NT and Windows 2000* describes in detail how to access set up sample stores in VisualAge for Java. The basic steps are:

1. Launch the WebSphere Commerce Administration Console by entereing the URL:

```
http://localhost:8080/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=adminconsole.AdminConsoleLogon
```

Be sure to use a supported Web browser when starting the Administration Console.

2. Log in using user wcsadmin with password wcsadmin.
3. Change the wcsadmin password.
4. Open the original browser window.

Important: Do not open a new browser window, but use the original window. This is important as the original browser window has the cookie of an administrator.

5. Publish the contract data. Enter the URL:

```
http://localhost:8080/wcs/contractPublish.html?storeId=10001
```

6. Access the sample store. Enter the URL:

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?storeId=10001&catalogId=10001&langId=-1
```

4.4.2 Create a sample store in VisualAge for Java

If you forgot to select the check box that would include a sample store to VisualAge for Java when installing WebSphere Commerce Studio, this section will guide you through how to manually create a store. A pre-requisite is that you know how to publish a store in Store Services, refer to Chapter 5, “Creating a store” on page 131.

Follow these steps to create a store in VisualAge for Java:

1. Go to the following URL to access the Store Services:
<http://localhost:8080/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=devtools.Logon>
2. Enter Store Services using the wcsadmin userID and publish a store.
3. When published, you will need to modify the viewreg as well as the urlreg tables hence https url calls are not runnable in the WebSphere Test Environment.
4. As the WebSphere Test Environment does not support SSL, we need to modify the URL and VIEW registrations in the database. Open an SQL command line window and execute the following sql statements:
 - **update viewreg set https=0**
 - **update urlreg set https=0**
5. Open the WebSphere Test Environment Control Center and restart the Servlet Engine.
6. Go to the store homepage, but change the port to 8080 and the protocol to http.

Tip: If you do not know the URL-path to the store, it can be launched from Store Services or enter the following URL:
http://localhost:8080/webapp/wcs/stores/servlet/Store_folder/index.jsp where *Store folder* is the folder specified when you created the SAR file.

4.4.3 Maintenance

To maintain VisualAge for Java to be functioning to its maximum potential, it is important that you regularly defrag your hard drive where you keep the installation of VisualAge for Java, Enterprise Edition. To do this, start the Defrag application that is located by clicking **Start -> Programs -> Accessories -> System Tools -> Disk Defragmenter** and choose to analyze the drive where the components have been installed and, if necessary, defrag the volume.

It is also important to do backups of the repository and the workspace files every now and then. To do a backup, choose the following files/folders:

- ▶ c:\WCDev\VAJ\ide\repository\ivj.dat file
- ▶ c:\WCDev\VAJ\ide\repository\ivj.dat.pr folder
- ▶ c:\WCDev\VAJ\ide\ide.icx file (the workspace)

4.5 WebSphere Studio Application Developer

This section will provide you with guidance on how to install and setup WebSphere Studio Application Developer to be used for WebSphere Commerce V5.4 development.

Attention: Please remember that developing WebSphere Commerce Business Edition V5.4 applications with WebSphere Studio Application Developer V4 is not officially supported. These instructions worked for us in our particular redbook environment and with the product versions we describe, but they may not work in your environment. We recommend that you carefully weigh the benefits and risks of developing with an unsupported combination of tools before investing time and effort in attempting this setup. We expect further changes to occur in this area and recommend that you watch the official WebSphere Commerce support site at:

<http://www-3.ibm.com/software/webservers/commerce/support.html>

for any updates to the support status of developing with WebSphere Studio Application Developer.

We expect that possible future support for WebSphere Studio Application Developer may be implemented differently and with different product versions and configurations and than those we describe in this section.

- ▶ Introduction
- ▶ Preparing for migration
- ▶ Setup WebSphere Studio Application Developer
- ▶ Setup WebSphere Test Environment
- ▶ The second installation
- ▶ Developing in WebSphere Studio Application Developer

4.5.1 Introduction

WebSphere Studio Application Developer is one of the members of the WebSphere Studio family of application development products from IBM. It is optimized for J2EE application development and evolved from VisualAge for Java, Enterprise Edition and WebSphere Studio Advanced Edition.

More and more development teams are using WebSphere Studio Application Developer as a tool for developing front end assets as well as back-office logic. This section will provide guidance in the migration work necessary to move from a VisualAge for Java and WebSphere Studio development environment to using WebSphere Studio Application Developer.

The advantages of using WebSphere Studio Application Developer as a development tool are:

- ▶ When the WebSphere Studio Application Developer environment has been properly setup, it will enable you to test your applications at a fraction of the time spent when testing in VisualAge for Java.
- ▶ WebSphere Studio Application Developer is a J2EE product and hence will the development environment follows the structure of the production environment. This will ease the deployment process.

To get a more detailed introduction to WebSphere Studio Application Developer, refer to the redpaper *An Introduction to IBM WebSphere Studio Application Developer* (REDP0414) or view the free online course *WebSphere Studio Application Developer - Presentations and labs* at:

<http://www7b.software.ibm.com/wsdd/library/presents/AppDevTraining.html>

Installation references

To avoid confusion caused by constantly referring to the installation paths of our development set up we list the installation directories we used in Table 4-1 so you can easily adapt the instructions to follow your system setup.

Attention: If you specify another path to a folder or file than specified in the installation or set up procedures of WebSphere Studio Application Developer in this redbook, be sure to specify the path without whitespaces. Use 8.3 DOS format in these path statements. At the command prompt, use `dir /x` to get the folder name listed in a 8.3 DOS name.

The installation instructions will assume that you have installed the WebSphere Commerce default instance named demo as well as the VAJ_Demo instance during the WebSphere Commerce Studio installation. This is because it is easier to refer to the different instances. You may of course name the instances to whatever name you prefer as long as it will follow the naming conventions of an instance name.

Table 4-1 Application paths

Application	Path
WebSphere Studio Application Developer	c:\WCDev\wsad
WebSphere Commerce Developer Edition	c:\WCDev\CommerceServerDev

Preparing for migration

This section will assist you to assure that you meet the pre-requisites and help you through the procedure of creating all necessary output for the WebSphere Studio Application Developer setup. It contains the following sections:

- ▶ Prerequisites
- ▶ Export the enterprise application
- ▶ Export EJBs from VisualAge for Java

Prerequisites

To be able to migrate to WebSphere Studio Application Developer there are a few pre-requisites that needs to be fulfilled.

- ▶ WebSphere Commerce Studio V5.4 must be installed on the development machine. You can use either the Professional or Business Edition of WebSphere Commerce Studio.
- ▶ VisualAge for Java, Enterprise Edition, must have been properly setup accordingly to 4.4.1, “Configure VisualAge for Java, Enterprise Edition” on page 90.
- ▶ The database used in the WebSphere Test Environment of VisualAge for Java, must have the same schema as used by WebSphere Commerce. By default, this will be true so the VAJ_Demo database uses the same schema as the mall database.
- ▶ WebSphere Commerce must be installed.
- ▶ A WebSphere instance running within WebSphere Application Server must have been created.

- ▶ Hardware requirements are the same as for WebSphere Commerce Studio, but with an additional of 1 GB disk space added for the WebSphere Studio Application Developer installation.

Note: The WebSphere Commerce installation and the WebSphere Commerce Studio installation do not necessarily need to be installed on the same machine.

Export the enterprise application

This section will guide you in how to export the current enterprise application of WebSphere Commerce. The export will produce an EAR file named `WC_Enterprise_App_instance.ear` (*instance* is the name of the WebSphere Commerce instance) and will later be used when setting up WebSphere Studio Application Developer. Follow these steps to export the enterprise application:

1. Start WebSphere Application Server Administration Console by clicking **Start -> IBM WebSphere -> Application Server 4.0 -> Administrator's Console**.
2. When the WebSphere Advanced Administrative Console appears, expand **WebSphere Administrative Domain -> Enterprise Applications**.
3. Right-click the WebSphere Commerce Enterprise Application - demo, and select **Export Application**.
4. When the Export Application window appears, enter a directory to export the application to. For example `c:\temp`.
5. Click **OK**.

Export EJBs from VisualAge for Java

To be able to get the public EJBs working in WebSphere Studio Application Developer, you will need to export the EJBs from VisualAge for Java to get the source-code into WebSphere Studio Application Developer at a later stage in the migration process. To export the EJBs, complete the following steps:

1. Start VisualAge for Java by clicking **Start -> Programs -> IBM VisualAge for Java for Windows V4.0 -> IBM VisualAge for Java**.
2. Go to the EJB page in VisualAge for Java.
3. Right-click on the **WCSActrlEJBGroup**, and click **Export -> EJB 1.1 JAR**.
4. Select a temporary folder to store the exported EJB and give it the name of the EJB, for example `c:\temp\EJBs\WCSActrlEjbGroup.jar`.
5. Ensure that the `.java` check box is selected and the window should look something like in Figure 4-6.

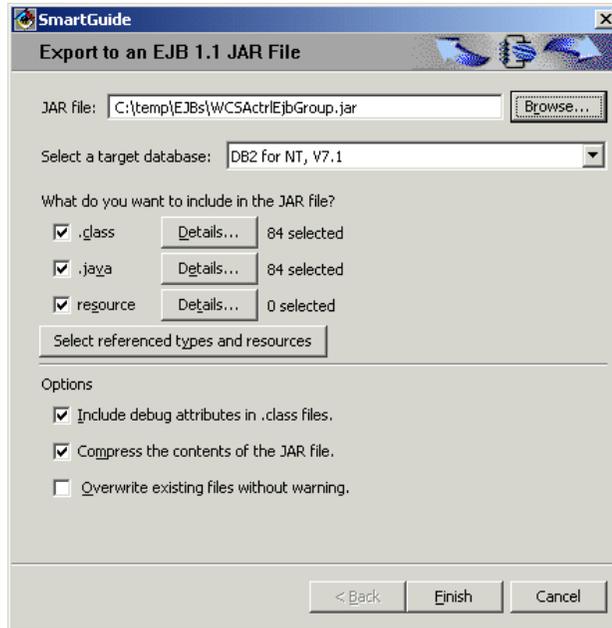


Figure 4-6 Export EJB

6. Click **Finish**.
7. When VisualAge for Java is finished with the export of the EJB, repeat step 3-6 for all other public EJBs. These are:
 - WCSActrIEJBGroup
 - WCSApproval
 - WCSAuction
 - WSCatalog
 - WCSCommon
 - WCSContract
 - WSCoupon
 - WCSFullfillment
 - WCSInventory
 - WCSMessageExtensions
 - WCSOrder
 - WCSOrderManagement
 - WCSOrderStatus
 - WCSPayment
 - WCSPVCDevices
 - WCSTaxation
 - WCSUser

- WCSUserTraffic
- WCSUTF

4.5.2 Setup WebSphere Studio Application Developer

This section will guide you through the process of setting up the development environment in WebSphere Studio Application Developer. Please read the instructions carefully. The following steps are:

- ▶ Install WebSphere Studio Application Developer
- ▶ Apply e-fixes
- ▶ Import the enterprise application
- ▶ Import public EJBs
- ▶ Verify EJB meta data
- ▶ Switch to the development instance
- ▶ Add Web aliases
- ▶ Browse to Store Services

Install WebSphere Studio Application Developer

To install WebSphere Studio Application Developer, complete the following steps:

1. Insert the WebSphere Studio Application Developer V4.03 CD into your CD-ROM drive.
2. When the IBM WebSphere Studio Application Developer welcome window appears, click **Next**.
3. When the License Agreement window appears, read the license agreement carefully and if you comply to the agreement click **Next**.
4. When the Destination Folder window appears, enter c:\WCDev\wsad and click **Next**.
5. When the Primary User Role window appears, select **J2EE Developer** and click **Next** to continue.
6. When the Select Version Control Interface window appears, select **CVS** and click **Next**.
7. When the Ready to Install the Program window appears, click **Install**.
8. When the Installshield Wizard Complete window appears, the installation is complete. Click **Finish** to close the window.

If any problems would arise, refer to the *WebSphere Studio Application Developer Installation Guide* that comes with the installation CD.

Install plugins

We need to install some extra plugins for WebSphere Studio Application Developer to work with WebSphere Commerce. The plugins are:

1. Zip creation plugin

The redbook additional material contains the file `com.ibm.sample.zip.creation(delta+prefix).zip`.

Extract the contents of this zip file into `<WSAD>\plugins`.

2. WebSphere Commerce copyright plugin.

The redbook additional material contains the file `com.ibm.commerce.plugins.wcs.copyright_1.0.2.zip`.

Extract the contents of this zip file into `<WSAD>\plugins`

Apply e-fixes

To get WebSphere Studio Application Developer to work properly for WebSphere Commerce development, you need to apply some e-fixes to the installation. This section will go through all the steps necessary to get the environment ready.

The fixes needed are:

1. WebSphere Application Server, Single Server Edition fixpack 3 available from:

`ftp://ftp.software.ibm.com/software/websphere/appserv/support/fixpacks/was40/fixpack3/Windows/was40_aes_ptf_3.zip`

To install this fixpack:

- a. Extract the contents of the zip file to a temporary directory
- b. Run the `install.bat` command file.
- c. When prompted to update the application server answer yes
- d. When asked to enter the application server home specify the WebSphere Studio Application Developer test environment JRE directory:
`<WSAD>\plugins\com.ibm.etools.websphere.runtime`
- e. When asked to enter the application server JDK specify the WebSphere Studio Application Developer server runtime directory:
`<WSAD>\plugins\com.ibm.etools.server.jdk`
- f. When prompted to update the JDK answer yes
- g. Answer no when asked whether to update the iPlanet Web server and the also when asked to update the IBM Http Server
- h. Answer yes to use the application server logs directory
- i. Answer yes to place backups in the application server home directory

2. WebSphere Studio Application Developer V4.03 APAR JR17012 EJB editor refresh problems available from:
<http://www-1.ibm.com/support/manager.wss?rs=0&rt=0&org=SW&doc=4001301>
To install this fix, extract the contents of the zip file to a temporary directory and run the setup.exe
3. WebSphere Studio Application Developer V4.03 APAR JR17026 errors when editing the source code of JSP files available from:
<http://www-1.ibm.com/support/manager.wss?rs=0&rt=0&org=SW&doc=4001298>
To install this fix, extract the contents of the zip file to a temporary directory and run the setup.exe
4. WebSphere Studio Application Developer V4.03 APAR JR17027 numerous fixes for EJB development available from:
<http://www-1.ibm.com/support/manager.wss?rs=0&rt=0&org=SW&doc=4001297>
To install this fix, extract the contents of the zip file to a temporary directory and run the setup.exe

Enable the IBMJCE security provider

1. Edit the file <WSAD>jre\lib\security\java.security file and modify the list of security providers to include IBMJCE. The provider list should be as in Example 4-2

Example 4-2 Java security providers

```
security.provider.1=sun.security.provider.Sun  
security.provider.2=com.ibm.crypto.provider.IBMJCA  
security.provider.3=com.ibm.crypto.provider.IBMJCE
```

2. Edit the file
<WSAD>plugins\com.ibm.etools.server.jdk\jre\lib\security\java.security file and modify the list of security providers to include IBMJCA. The provider list should be as in Example 4-3.

Example 4-3 Java security providers for test server

```
security.provider.1=sun.security.provider.Sun  
security.provider.2=com.ibm.crypto.provider.IBMJCE  
security.provider.3=com.ibm.jsse.JSSEProvider  
security.provider.4=com.ibm.crypto.provider.IBMJCA
```

Import the enterprise application

To import the enterprise application that you previously exported from the WebSphere Application Server follow these steps:

1. Start WebSphere Studio Application Developer by clicking **Start -> Programs -> IBM WebSphere Studio Application Developer -> IBM WebSphere Studio Application Developer**.
2. From the main menu, select **File -> Import**.
3. When the Import wizard appears, select **EAR file** and click **Next**.
4. When the EAR Import window appears, click **Browse** and select the EAR file named WC_Enterprise_App_demo.ear that you previously exported from the WebSphere Application Server. The window should look like Figure 4-7.

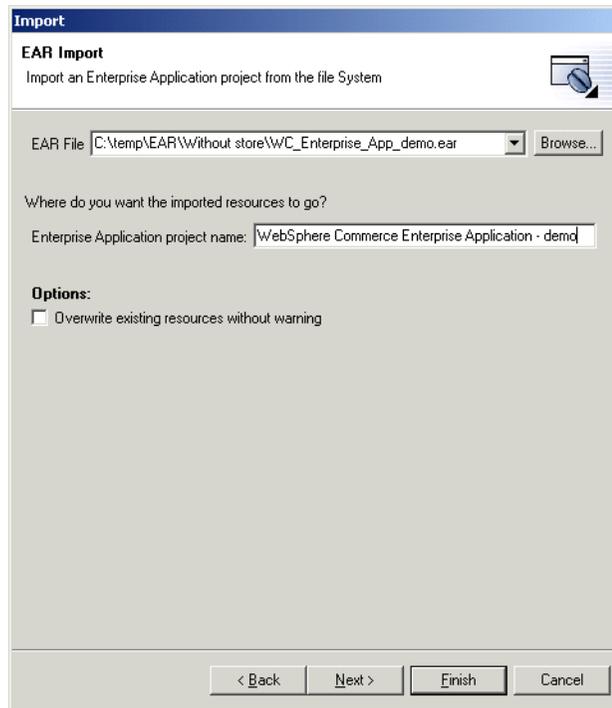


Figure 4-7 EAR Import

5. Click **Finish** to start the import.

Import public EJBs

To import the public EJBs, which you previously exported from VisualAge for Java, you need to do the following:

1. From the main menu in WebSphere Studio Application Developer, select **Windows -> Preference**.

2. When the Preference window appear, select **Workbench** and uncheck **Perform build automatically on resource modification**.
3. Click **Apply** and then click **OK** to close the window.
4. From the main menu in WebSphere Studio Application Developer, select **File** -> **Import**.
5. When the Import wizard appears, select **EJB JAR file** and click **Next**.
6. When the EJB Import window appears, click **Browse** and select the **WCSActrIEJBGroup.jar** file that you previously exported from VisualAge for Java.
7. Select the corresponding EJB project to be the destination of the imported files, select the **Overwrite existing resources without warning** check box. The window will look like Figure 4-8.

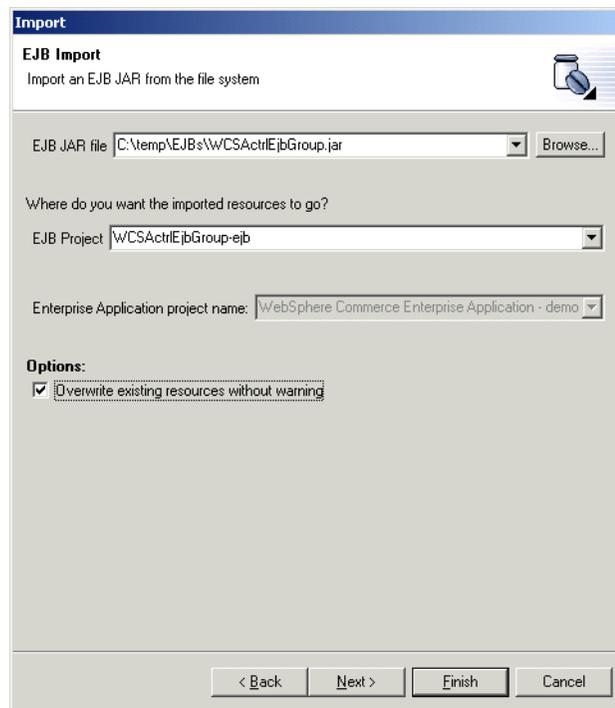


Figure 4-8 EJB Import

8. Click **Finish**.
9. When the EJB name collision window appears, just click **Yes** to continue.
10. Repeat procedure 4-9 for all the public EJBs. These are the following:

- WCSActrIEJBGroup
 - WCSApproval
 - WCSAuction
 - WSCCatalog
 - WCSCommon
 - WCSContract
 - WSCCoupon
 - WCSFullfillment
 - WCSInventory
 - WCSMessageExtensions
 - WCSOrder
 - WCSOrderManagement
 - WCSOrderStatus
 - WSCPAYMENT
 - WCSPVCDevices
 - WCSTaxation
 - WCSUser
 - WCSUserTraffic
11. Open Windows Explorer and browse to the workspace directory of WebSphere Studio Application Developer, currently c:\WCDev\wsad\workspace.
 12. Go through the EJB projects that you imported the public EJBs to and remove all *.imported_classes.jar files. For example for WCSActrIEJBGroup-ejb project, remove the WCSActrIEjbGroup-ejb.imported_classes.jar file.

Verify EJB meta data

This section will help you through the process of verifying the module dependencies as well as the JNDI binding of the EJBs. Complete the following steps to verify the meta data of each of the EJB projects:

1. Make sure that the J2EE perspective is selected in WebSphere Studio Application Developer.

Tip: In WebSphere Studio Application Developer, the perspective currently in use is shown on the title bar of the Workspace.

2. Expand the **EJB Modules**, and right-click on the first EJB module (project) and select **Open With -> EJB Extension Editor**.
3. When the EJB Extension Editor opens, click on the **Bindings** page and expand the module.

- For each of the EJBs in the module, verify that the JNDI name does not contain the prefix of the WebSphere Commerce Server instance. See example in Figure 4-9.

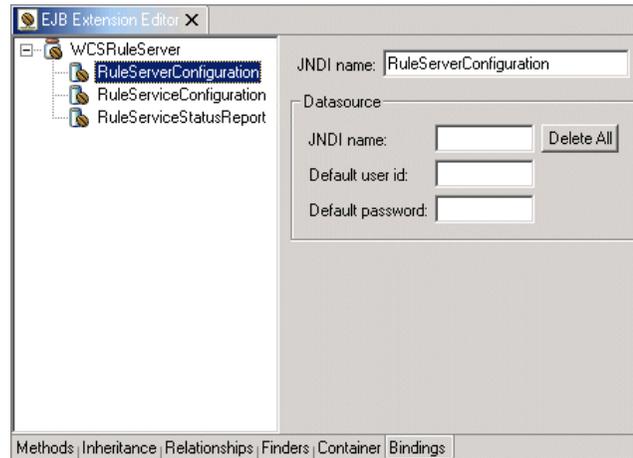


Figure 4-9 EJB Extension Editor

- Close the EJB Extension editor.
- You will be prompted to save the changes made in the EJB Extension Editor. Click **Yes** to continue.
- Right-click the same module again and select **Edit Module Dependencies**.
- When the Module Dependencies Dialog window appears, verify that `lib/wcsejbimpl.jar`, `lib/xml4j.jar` and all other EJB modules have been selected.
- Click **Finish**.

Attention: The Module Dependencies Dialog displays the deployed EJB settings. I.e. if you open the dialog again, the original settings will be displayed.

- Right-click the same EJB module again and select **Generate -> Deploy and RMIC Code**.
- When the Generate Deploy and RMIC Code window appears, click **Select all** and click **Finish** to complete.
- Repeat the procedure for all the EJB modules.

Switch to the development instance

Currently is the environment setup to be using the demo instance (as the settings were exported with the EAR file). This section will guide you through the changes necessary to start using the VAJ_Demo instance instead.

Note: Do not confuse this with the naming of the instances. The changes made will refer to the development instance, but it will keep the JNDI name of the production environment.

1. Open the
c:\WCDev\CommerceServerDev\instances\VAJ_Demo\xml\VAJ_Demo.xml
file in a text-editor
2. Search for the names in the VAJ_Demo file and change the values to what has been specified in Table 4-2.

Table 4-2 Changes in VAJ_Demo.xml

Name	Value
DatasourceName	WebSphere Commerce DB2 DataSource demo
StoresDocRoot	C:\WCDev\wsad\workspace\wcstores
StoresPropertiesPath	webApplication\WEB-INF\classes
StoresWebPath	webApplication

Tip: When deploying a new store in Store Services, the store properties will be in a subdirectory of the classes directory as specified in Table 4-2. When rebuilding the webmodule, the classes directory will be emptied and created from the source folder, hence copy the new subdirectory to the source folder.

3. Save and close the file.
4. In WebSphere Studio Application Developer go to the J2EE perspective.
5. Go to the J2EE view. If it is not on the workbench, go to the main menu; click **Perspective -> Show View -> J2EE View**.
6. Expand the Web Modules, right-click on the **WCS Stores** module and select **Open With -> Web.xml editor**.
7. When the web.xml has opened, click on the **Servlets** tab.
8. Click **Initialization**.

9. When the Initialization Parameters window appears, modify the settings according to the information in Table 4-3.

Table 4-3 WCS Stores Initialization parameters

Initialization Parameter	Value
configfile	C:\WCDev\COMMER~1\INSTAN~1\VAJ_Demo\xml\VAJ_Demo.xml
instancename	VAJ_Demo
webpath	/webapp/wcs/stores

10. When you are done with the changes, click **OK**.
11. On the main menu, select **File -> Save web.xml** to store the changes.
12. Close the web.xml window.
13. In the J2EE view, right-click on the **WCS Tools** module and select **Open With -> Web.xml editor**.
14. When the web.xml has opened, click on the **Servlets** tab.
15. Click **Initialization**.
16. When the Initialization Parameters window appears, modify the settings according to information in Table 4-4.

Table 4-4 WCS Tools Initialization parameters

Initialization Parameter	Value
configfile	C:\WCDev\COMMER~1\INSTAN~1\VAJ_Demo\xml\VAJ_Demo.xml
instancename	VAJ_Demo

17. When you are done with the changes, click **OK**.
18. On the main menu, select **File -> Save web.xml** to store the changes.
19. Close the web.xml window

Apply ivjfix

To get the environment properly set up, you need to add a fix that is to be loaded into the servlet context of the WCS Store and WCS Tools modules. This is how you go about adding the fix to WebSphere Studio Application Developer:

1. On the main menu, select **File -> Import**.
2. When the import wizard appears, select **File system** and click **Next**.

3. When the File System window appears, click the first **Browse** button to specify a directory and select **c:\WCDev\CommerceServerDev\lib**. The lib directory will appear in one of the list boxes.
4. Click on the lib folder.

Attention: Do not select the check box of the lib folder.

5. A listing of files appears in one of the boxes. Select the check box of the **ivjfix.jar** file.
6. Specify the destination of ivjfix.jar by clicking on the second **Browse** button and select the **WebSphere Commerce Enterprise Application - demo/lib** directory.
7. Click **Finish**.
8. Go to the J2EE perspective and go to the Navigator view. If you cannot find the Navigator view in the perspective, on the main menu select **Perspective -> Show View -> Navigator**.
9. Right-click the **wcstores** project and select **Edit Module Dependencies**.
10. Verify that the check box for lib/ivjfix.jar file is selected as a dependent JAR for the module. Click **Finish**.
11. Right-click the **wcstores** project and select **Properties**.
12. Go to the **Java Build Path** page.
13. Click on the **Order** tab.
14. Scroll and select the **ivjfix.jar** in the list box.
15. Click on **Up** or **Down** button to change the order of the ivjfix.jar file to position the file just above the line with the ivjeb35.jar file as in Figure 4-10.

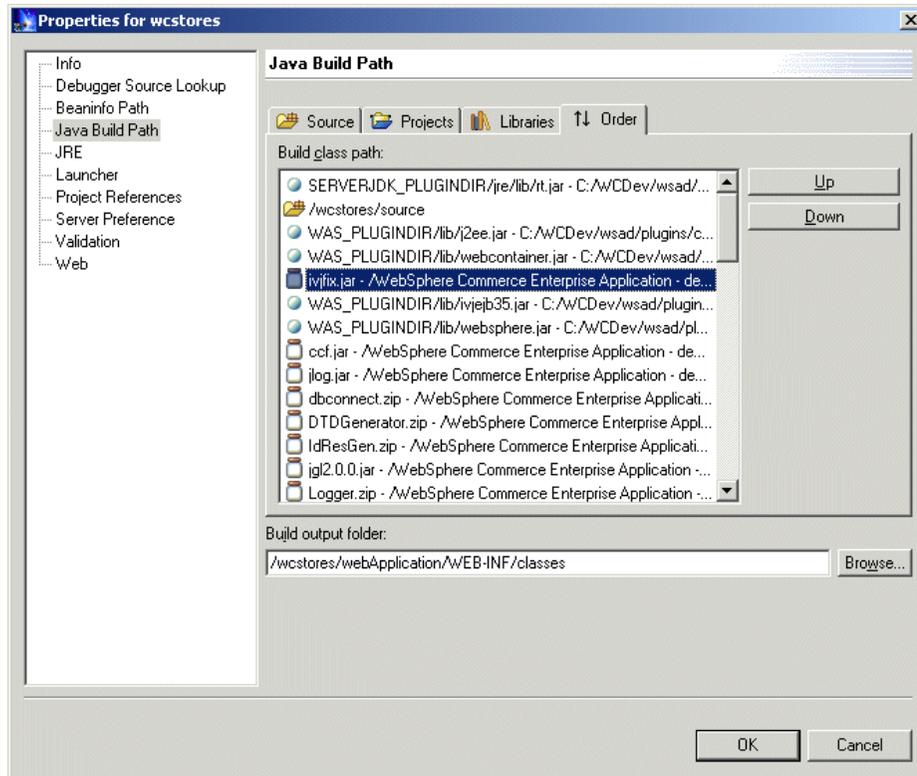


Figure 4-10 Position of the ivjfix.jar file

16. Click **OK**.

17. Repeat step 9 to 16 for the wctools project.

Add Web aliases

As Web aliases are not supported inside the WebSphere Test Environment of WebSphere Studio Application Developer, you need to apply a workaround to the wcstores as well to the wctools project. Follow these steps:

1. On the main menu of WebSphere Studio Application Developer, select **File -> New -> Web Project**.
2. When the Define the Web Project window appear, enter **wcstorealias** as project name, verify that **WebSphere Commerce Enterprise Application - demo** is selected as Enterprise Application project name and click **Finish**.
3. On the main menu of WebSphere Studio Application Developer, select **File -> Import**.

4. When the import wizard opens, select **WAR file** as a source to be imported. Click **Next**.
5. When the Identify the WAR File and Import Options window appear, click **Browse** to specify the location of the **wcstorealias.war** file that is provided as additional material to this redbook.

Author Comment: wcstorealias.war is to be supplied with the redbook.

6. Select the **wcstorealias** as the Web project.
7. Enter **/wcsstore** as the value for the Context Root.
8. Make sure that **WebSphere Commerce Enterprise Application - demo** is selected as the Enterprise Application project name.
9. Click **Next** until the Define Java Build Settings window appears.
10. Click on the **Libraries** tab.
11. Ensure that you have the following in your build path:
 - WAS_PLUGINDIR/lib/nls.jar
 - WAS_PLUGINDIR/lib/utls.jar
 - WAS_PLUGINDIR/lib/webcontainer.jar
 - WAS_PLUGINDIR/lib/websphere.jar

Note: If you do not have all these JARs, click **Add Variable** and specify the Variable Name and Path extension with the use of the Browse buttons.

12. Click **Finish**.
13. You will be prompted whether or not you want to overwrite the existing web.xml file. Click **Yes**.
1. On the main menu of WebSphere Studio Application Developer, select **File -> New -> Web Project**.
2. When the Define the Web Project window appear, enter **wctoolsa1ias** as project name, verify that **WebSphere Commerce Enterprise Application - demo** is selected as Enterprise Application project name and click **Finish**.
3. On the main menu of WebSphere Studio Application Developer, select **File -> Import**.

4. When the import wizard opens, select **WAR file** as a source to be imported. Click **Next**.
5. When the Identify the WAR File and Import Options window appear, click **Browse** to specify the location of the **wctoolsalias.war** file that is provided as additional material to this redbook.

Author Comment: wctoolsalias.war is to be supplied with the redbook.

6. Select the **wctoolsalias** as the Web project.
7. Enter **/wcs** as the value for the Context Root.
8. Make sure that **WebSphere Commerce Enterprise Application - demo** is selected as the Enterprise Application project name.
9. Click **Next** until the Define Java Build Settings window appears.
10. Click on the **Libraries** tab.
11. Ensure that you have the following in your build path:
 - WAS_PLUGINDIR/lib/nls.jar
 - WAS_PLUGINDIR/lib/utls.jar
 - WAS_PLUGINDIR/lib/webcontainer.jar
 - WAS_PLUGINDIR/lib/websphere.jar
12. Click **Finish**.
13. You will be prompted whether or not you want to overwrite the existing web.xml file. Click **Yes**.

4.5.3 Setup WebSphere Test Environment

To be able to test your code, you will need to setup an test-environment in WebSphere Studio Application Developer. Complete the following steps to setup the test environment.

- ▶ Create a server project
- ▶ Create an instance and configuration
- ▶ Associating the project to the server configuration
- ▶ Modify the server configuration
- ▶ Modify the server instance

- ▶ Start the server
- ▶ Define virtual host for the server instance

Tip: Prior to start working in WebSphere Studio Application Developer, refer to the technical article *Optimizing Multi-Project Builds Using Dependent Project JARs in WebSphere Studio Application Developer* to get some performance related advice.

http://www7b.software.ibm.com/wsdd/library/techarticles/0204_searle/searle.html

Create a server project

1. From the File menu, select **New -> Project**.
2. When the New Project window appears, select **Server** in the left list and **Server Project** in the right list. Click **Next**.
3. When the Create New Server Project window appears, enter a project name, for example WC Dev Server, and click **Finish**.
4. The server project is created and Server perspective will open.

Create an instance and configuration

To create a server instance and a server configuration, complete the following steps:

1. In the Navigator view, right-click the server project that you just created, and select **New -> Server Instance and Configuration**.
2. When the wizard opens enter for Server name **Dev WTE**, make sure that server project created is the folder and select **WebSphere Servers -> WebSphere V4.0 Test Environment** as server instance type. The window will look like Figure 4-11.

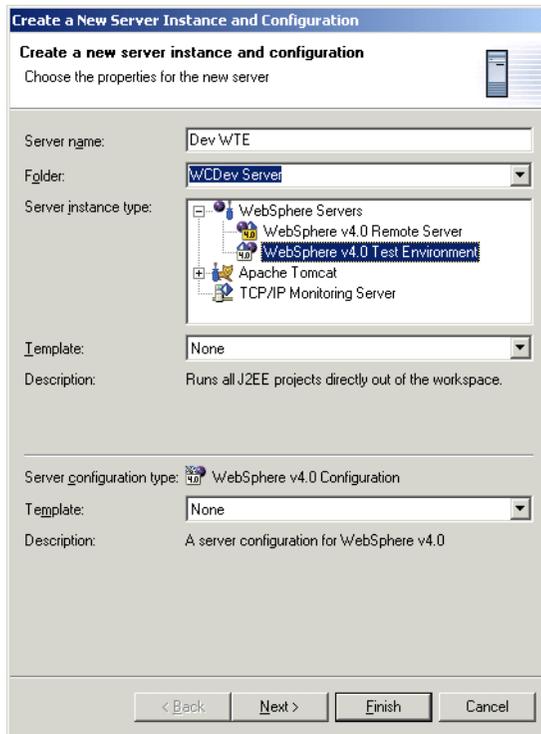


Figure 4-11 Create a New Server Instance and Configuration

3. Click **Finish**. The new server configuration folder and the new server instance appear under the project folder in the Navigator view.

Associating the project to the server configuration

Once you have created a server configuration and instance, you will need to associate the Enterprise Application with the server configuration:

1. If the Server Configuration view is not on the workbench, on the main menu bar, click **Perspective -> Show View -> Server Configuration**.
2. Expand the Server Configurations folder and right-click on the server configuration previously created, i.e. WCDev WTE, and select **Add project -> WebSphere Commerce Enterprise Application - demo**.
3. The association is being made as you can now expand the server configuration to find the Enterprise Application.

Modify the server configuration

To be able to run the WebSphere Commerce in WebSphere Studio Application Developer you need to modify the server configuration. Follow these steps to complete the server setup:

1. Go to the Servers view. If you cannot find it in the perspective, on the main menu select **Perspective -> Show View -> Servers**.
 2. Expand Server Configurations, right-click the server and select **Open**.
 3. The server configuration opens. Click on the **General** tab.
 4. Select **APPLICATION** as Module visibility and check **Enable administration client**.
 5. Click on the **Web** tab.
 6. Make sure that the Enable session manager is not checked, Enabled URL rewrite is checked and that Enable cookies is checked.
 7. Click on the **Data source** tab.
 8. Click **Add** on the button next to the list of datasources.
 9. When the Add a Data Source window appears, enter the following information:
 - Name = **WebSphere Commerce DB2 DataSource demo**
 - JNDI name = **jdbc/WebSphere Commerce DB2 DataSource demo**
 - Database name = **VAJ_Demo**
 - For Default user ID/password, enter the user ID and password you use to access the VAJ_Demo database.
- Important:** Make sure to specify the database created at the setup of WebSphere Commerce Studio installation.
10. Click **OK**.
 11. Click on the **Ports** tab.
 12. Next to the HTTP transport list, click **Add**.
 13. When the Add a HTTP Transport window appears, enter * as Host name and **80** as Port. Verify that Enable SSL is not checked and that External is checked. Click **OK**.
 14. Next to the HTTP transport list, click the **Add**.
 15. When the Add a HTTP Transport window appears, enter * as Host name and **443** as Port. Verify that Enable SSL is checked and that External is checked. Click **OK**. The server configuration should now look like Figure 4-12.

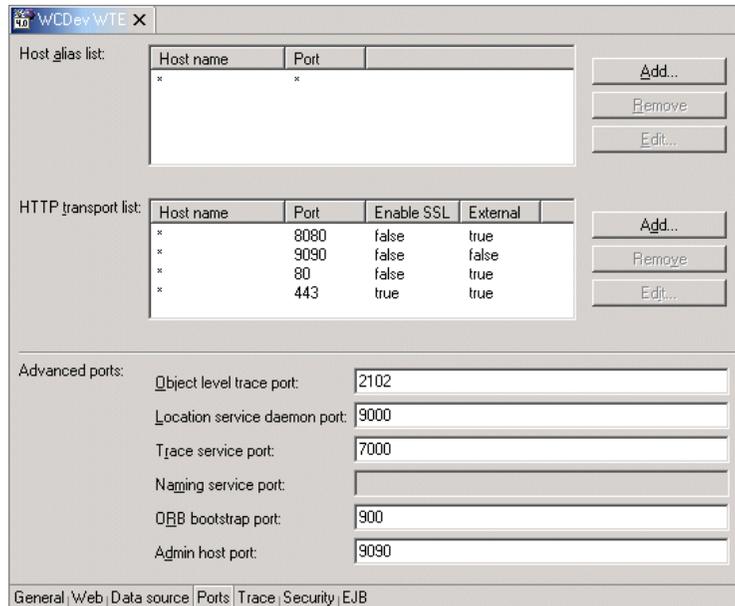


Figure 4-12 Server ports settings

16. Close the window. You will be prompted to save the changes. Click **Yes**.

Modify the server instance

The JVM that is to run WebSphere Commerce in WebSphere Studio Application Developer needs to get some system properties passed at start-time. Follow these steps to configure your instance:

1. Go to the Servers view. If you cannot find it in the perspective, on the main menu select **Perspective -> Show View -> Servers**.
2. Expand Server Instances, right-click the server and select **Open**.
3. Click on the **Environment** tab.
4. Click **Add** to add a new system property.
5. As Name enter **com.ibm.wca.logging.configFile** and as value enter **C:\WCDev\CommerceServerDev\xml\loader\WCALoggerConfig.xml**.

Attention: Make sure that you enter the path to the CommerceServerDev folder, and not to the CommerceServer folder.

6. Click **OK**.

- Repeat step 4 to 6 until you have all names and values listed in the box as specified in Table 4-5.

Table 4-5 System Properties

Name	Value
com.ibm.wca.logging.configFile	C:\WCDev\CommerceServerDev\xml\loader\WCALoggerConfig.xml
javax.rmi.CORBA.UtilClass	com.ibm.CORBA.iiop.Util
com.ibm.websphere.ejbcontainer.FbpkAlwaysReadOnly	true
com.ibm.ws.classloader.ejbDelegationMode	false
com.ibm.CORBA.iiop.noLocalCopies	true

- Click on the **Paths** tab.
- Click on the **Add Folder** button next to the Class Path list box (not the WebSphere specific class path list box).
- Scroll to and expand the **WebSphere Commerce Enterprise Application - demo** folder.
- Select the **properties** folder.
- Click **OK**. The properties folder will now be listed in the Class Path list box.
- Close the window. You will be prompted to save the changes. Click **Yes**.

Start the server

To start the server instance, follow these steps:

- Go to the Servers view. If you cannot find it in the perspective, on the main menu select **Perspective -> Show View -> Servers**.
- Right-click on the server instance and select **Publish**.
- When the publishing is complete, close the popup window by clicking on **OK**.
- Right-click on the server instance and select **Start**.
- A Console view opens and you can follow the startup process of the instance. It is ready when the final line in the console view reads

Define virtual host for the server instance

After the first time the server instance is started, it is necessary to define the virtual hosts to be used for the Enterprise Application. Follow these steps:

1. If the Server Configuration view is not on the workbench, on the main menu bar, click **Perspective -> Show View -> Server Configuration**.
2. Expand Server Instances and right-click on your server instance and select **Run administrative client**.
3. A browser window opens and soon you are prompted for a user ID. The user ID is only for logging purposes and thus you might enter anything. Click **Submit**.
4. Expand the Node tab until you see your server under Enterprise Applications.
5. Click on the **Modify virtual host mapping** link.
6. When the Application Installation Wizard appears, verify that **default_host** is selected as Virtual Host Name for all Web modules. Click **Next** to continue.
7. To confirm the changes, click **Finish**.
8. Save the changes made by clicking **Save** on the toolbar and then click **OK** when the Save Configuration page appears.
9. Close the window.
10. Go to the Servers view. If you cannot find it in the perspective, on the main menu select **Perspective -> Show View -> Servers**.
11. Right-click on the server instance and select **Stop**.
12. Follow “Start the server” on page 118, to publish and start the server again to make the virtual host settings effective.

Browse to Store Services

You are now ready to browse to the Store Services and create a new store. Follow these steps to access and setup the store in WebSphere Studio Application Developer:

1. Open a browser window and type the URL:
<https://localhost/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=devtools.Logon>
2. After a few seconds, the logon page to the Store Services will appear. Enter Store Services using the wcsadmin userID and publish a store.

Author Comment: Refer to another chapter below.

Tip: Refer to , on how to create and publish a store.

Note that in comparison to the WebSphere Test Environment in VisualAge for Java, you are able to do HTTPS requests due the e-fixes you applied and to the set up you have just completed.

4.5.4 The second installation

The migration work from VisualAge for Java to WebSphere Studio Application Developer involves a lot of work, and it is only necessary to be done once. This section will provide a short description on how to setup a second WebSphere Commerce development environment with WebSphere Studio Application Developer.

Installation

The installation process of a second machine with WebSphere Commerce and WebSphere Studio Application Developer comprise the following steps:

- ▶ Save the workspace
- ▶ Simulate a VisualAge for Java installation
- ▶ Install WebSphere Commerce Studio
- ▶ Install WebSphere Studio Application Developer
- ▶ Post-install set up

Save the workspace

We recommend that you copy the entire workspace from the original machine, as it is the quickest and most efficient way to setup a second machine. By doing a workspace copy, you will get a configured test server as well as the complete enterprise application.

Prior to saving the workspace, remove the cached pages of the enterprise application. These can be found at *WebSphere Studio Application Developer\workspace\metadata\plugins\com.ibm.etools.server.tools\tmpX\cache\localhost\Default Server\WebSphere_Commerce_Enterprise_Application_-_demo*.

Simulate a VisualAge for Java installation

If you do not have VisualAge for Java installed on your system, and do not want to have it installed, you need to do a simulation of an VisualAge for Java installation to be able to install WebSphere Commerce Studio together with a database.

Complete these steps to set up a simulated VisualAge for Java installation:

1. On the machine used for the migration work, use the Windows Registry Editor to export the VisualAge for Java properties. These are found at: MY COMPUTER - HKEY_LOCAL_MACHINE - SOFTWARE - IBM - VisualAge for Java for Windows.

Attention: Inaccurate changes to the Windows registry can cause your computer to stop functioning properly. Do not do these steps if you do not know exactly what you are doing.

2. Copy the exported file onto the new machine and execute the file to import the registry settings.
3. Create the VisualAge for Java directory on the new machine at the same location as on the installed system. For example c:\WCDev\VAJ.

Tip: The exported registry file contains the directory information. Search for a parameter named Directory and you will find the installation directory of VisualAge for Java.

Install WebSphere Commerce Studio

Follow the instructions in “Install WebSphere Commerce Studio” on page 86, with the following exceptions:

- ▶ When the Select Components window appears, only select **Develop Store Back-Office Logic using VisualAge for Java** as a Development Type.
- ▶ Do not include a sample store with the installation.
- ▶ When you are to provide the Database information, you must provide the same userid and password as is being used on the original machine.

Install WebSphere Studio Application Developer

Follow the installation instructions provided in “Install WebSphere Studio Application Developer” on page 101 and apply the e-fixes as described in “Apply e-fixes” on page 102.

Post-install set up

To finish the installation set up, complete the following steps:

1. Remove the fake VisualAge for Java folder and registry settings.
2. Move the saved workspace from the original machine to the installation directory of WebSphere Studio Application Developer on the new machine.
3. Restart the machine.

Configuration

When the installation is complete, some verifications in WebSphere Studio Application Developer are necessary. Follow these steps:

- ▶ Update Enterprise Application configuration
- ▶ Update Web module settings, VAJ_Demo.xml and Server Instance
- ▶ Update server configuration

Update Enterprise Application configuration

The Web modules path configuration needs to be updated in the Enterprise Application configuration file, application.xml. Follow these steps:

1. Start WebSphere Studio Application Developer and go to the J2EE perspective.
2. Expand the Enterprise Applications in the J2EE view and double-click on **WebSphere Commerce Suite**.
3. If you installed WebSphere Studio Application Developer in a different directory relative to the root drive in comparison to the original machine, you will be prompted to auto correct the absolute path of the modules. Click **Yes**.
4. Save and close the application.xml file.

Update Web module settings, VAJ_Demo.xml and Server Instance

Follow the section “Switch to the development instance” on page 108, to get instructions on how to verify that the information passed to Web modules, as configuration parameters, are correct and how to configure the instance configuration file, VAJ_Demo.xml, to enable the instance to be used with WebSphere Studio Application Developer.

Verify that the path statement for the com.ibm.wca.logging.configFile system property in the server instance configuration is correct. Refer to Table 4-5 on page 118.

Update server configuration

You might need to update the server configuration. This is necessary if you installed WebSphere Studio Application Developer in a different directory relative to the root drive in comparison to the original machine. Follow these steps to complete the update of the server configuration:

1. Go to the Server perspective in WebSphere Studio Application Developer.
2. Verify that you got a server instance as well as a server configuration in the Server Configuration view. If you do not see the Server instance, do a Refresh from Local on the Server project.
3. In the Server Configuration view, right-click the server and select **Open**.

4. If any changes to the server configuration are necessary, you will be prompted to automatically have the paths updated. Click **Yes**.
5. Save and close the server configuration.

4.5.5 Developing in WebSphere Studio Application Developer

When developing commands and EJBs within the J2EE framework, you need to store the code in a location reachable for the Enterprise Application. We have found the easiest way to develop in WebSphere Studio Application Developer is to use a temporary directory to keep the application code in and which path has been added to the Application Servers classpath.

When the code is complete, we deploy the code as a JAR file and add it to the Enterprise Application.

As a short introduction in developing WebSphere Commerce components in WebSphere Studio Application Developer, we will in this section provide an example on how to create and test a very simple command. This section will not explain the coding done, as its intension is to provide guidance in how to create the code in WebSphere Studio Application Developer and how to test it.

The tutorial is divided in the following sections:

- ▶ Create an example
- ▶ Configure the WebSphere Test Environment
- ▶ Test the command

Create an example

Follow these steps to create an easy example in WebSphere Studio Application Developer:

1. Create a new Web project with the name WCDevelopment and in the Web Project Wizard add it to the Enterprise Application as shown in Figure 4-13. Click **Finish**.

The intension is that we will now use this created Web project for all WebSphere Commerce development. A Web module has now been added to the Enterprise Application, but if necessary it may be removed (and replaced) from the Enterprise Application Editor.

Tip: The Enterprise Application editor is accessible from the J2EE view. Right-click the Enterprise Application and select **Open With -> Application Editor**.

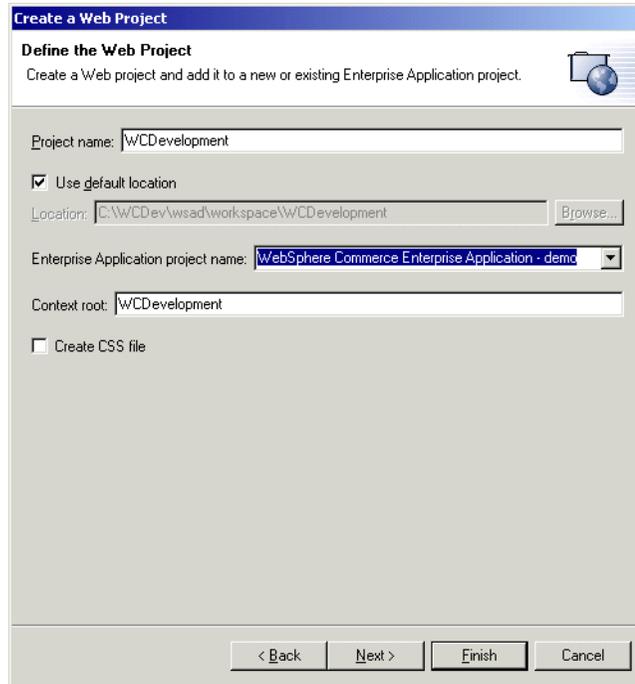


Figure 4-13 Create a Web Project

2. In order for the command we are about to create to be compiled, the module dependencies for the Web project needs to be updated. Right-click the project folder and select **Edit Module Dependencies**.
3. The Module Dependencies Dialog appears, and we select all the available dependent JAR files in the listing. Click **Finish**.

Of course you only need to add the dependant JAR files for your command, but as we plan to use the same Web project for comming WebSphere Commerce development we might as well add all to the project.

4. Go to java perspective and from the main menu select **File -> Import**.
5. When the Import wizard appears, select to import a ZIP file. Click **Next** to continue.
6. Browse to the additional material of the redbook and select the NewWsadContollerCmd.zip as the ZIP file to be imported. Refer to Figure 4-14 for the other settings in the import guide. Click **Finish**.

The code is now being imported to the WCDevelopment project on the workbench.

Author Comment: MyNewWsdControllerCmd.zip is additional material

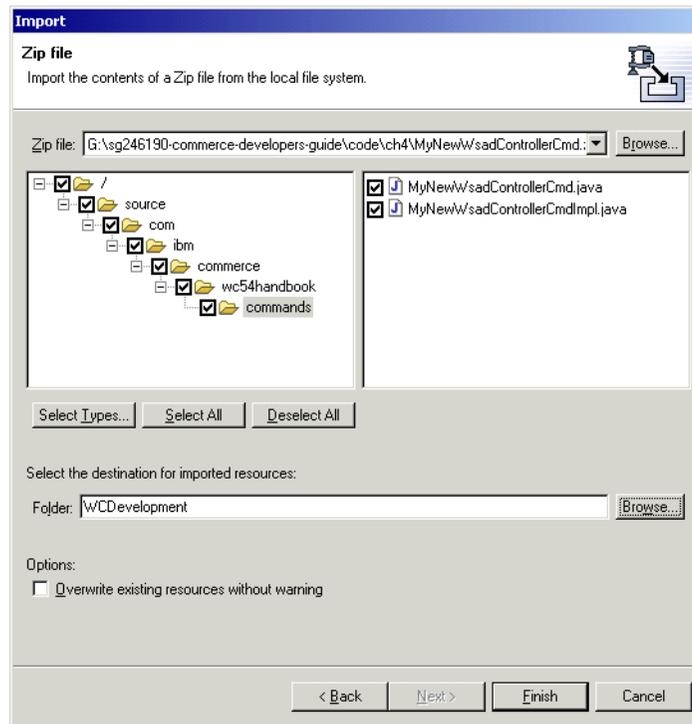


Figure 4-14 Import MyNewWsdContollerCmd

7. Now we will import the JSP page that is going to be the page called upon by the controller command. From the main menu, select **File -> Import**.

Author Comment: The JSP file is additional mtrl

8. When the Import wizard appears select to import a File system resource. Click **Next**.
9. Set the import properties as shown in asdf to import the WsadSample.jsp file to the webApplication directory of the wcstores project. Click **Finish** when ready.

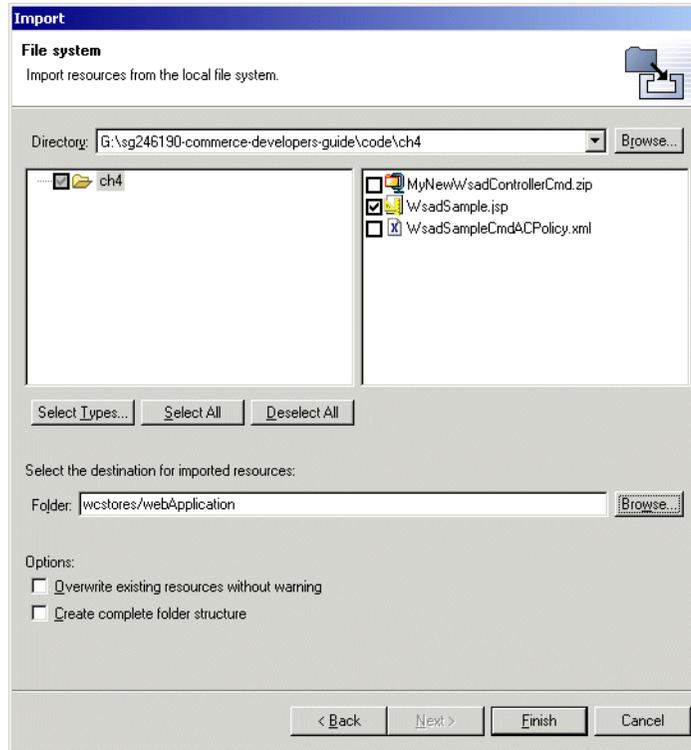


Figure 4-15 Import JSP file

10. To be able to test the command in WebSphere Test Environment we will have to register the command in the URLREG table and the view it is calling in the VIEWREG table. Run the following SQL statements to your VAJ_Demo database to get them registered:

- a.

```
insert into URLREG (URL,STOREENT_ID,INTERFACENAME,HTTPS,
DESCRIPTION,AUTHENTICATED) values ('MyNewWsdControllerCmd',0,
'com.ibm.commerce.wc54handbook.commands.MyNewWsdControllerCmd',0
,null,null)
```
- b.

```
insert into VIEWREG (VIEWNAME,DEVICEFMT_ID,STOREENT_ID,
INTERFACENAME,CLASSNAME,PROPERTIES,DESCRIPTION,HTTPS,LASTUPDATE)
values ('WsdSampleViewTask',-1,0,
'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl',
'docname=WsdSample.jsp',null,0,null)
```

11. To be able to run this command, we need to specify its command level access control. The policies have been specified in a file named

WsdSampleCmdACPolicy.xml and it is loaded by completing the following steps:

- a. Copy the XML file to c:\WCDev\CommerceServerDev\xml\policies\xml directory.
- a. Open a command prompt at where the WsdSampleCmdACPolicy.xml file is located.
- b. Issue the acpload command by executing the following statement:

```
c:\WCDev\CommerceServerDev\bin\acpload VAJ_Demo userID password  
WsdSampleCmdACPolicy.xml
```

Tip: View the generated file messages.txt to verify that the load has been successful. The last row should read: Number of records committed: 3.

Author Comment: WsdSampleCmdACPolicy.xml is additional mtrl

Configure the WebSphere Test Environment

In order for the new classes to be found, we need to modify the class path in the test server instance configuration. Follow these steps to complete the configuration:

1. Go to the Server perspective.
2. In the Server view, right-click the server instance and select **Open**.
3. Click on the **Paths** tab.
4. Click on **Add Folder** and select the /webApplication/WEB-INF/classes folder of the WCDevelopment project to add the folder to the class path. The configuration will look like Figure 4-16.

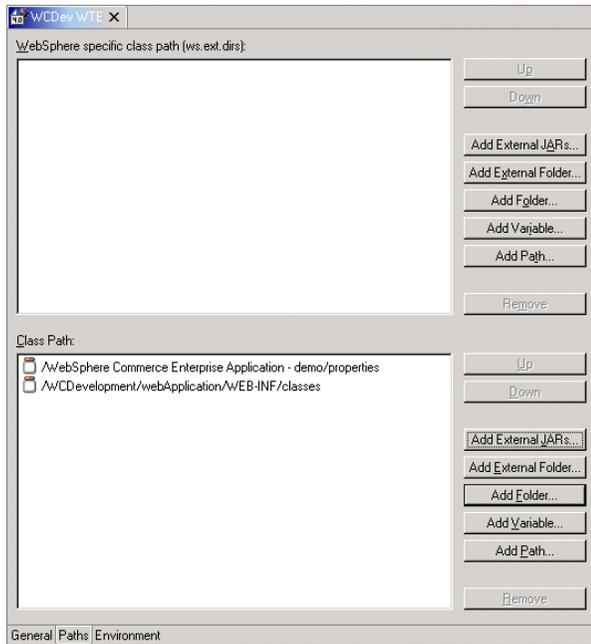


Figure 4-16 Add folder to class path

5. Close the configuration window, and when prompted click **Yes** to save the changes.

Test the command

In order to test the command, follow these steps:

1. Make sure that the WCDevelopment project has been built. This can be done by right-clicking the project folder and select build project.

Note: This is only necessary if you have disabled the Auto-build option in the Workbench preferences.

2. Go to the Servers view in the Server perspective.
3. Right-click on the server and select **Publish**. When the publishing is complete, click **OK** to close the window.
4. Right-click on the server and select **Start**. The start up procedure of the application server commence.
5. When the console output reads Server Default Server open for e-business, open a browser window and enter the following URL:

<http://localhost/webapp/wcs/stores/servlet/MyNewWsadControllerCmd>

After a few seconds the browser displays a page as shown in Figure 4-17.

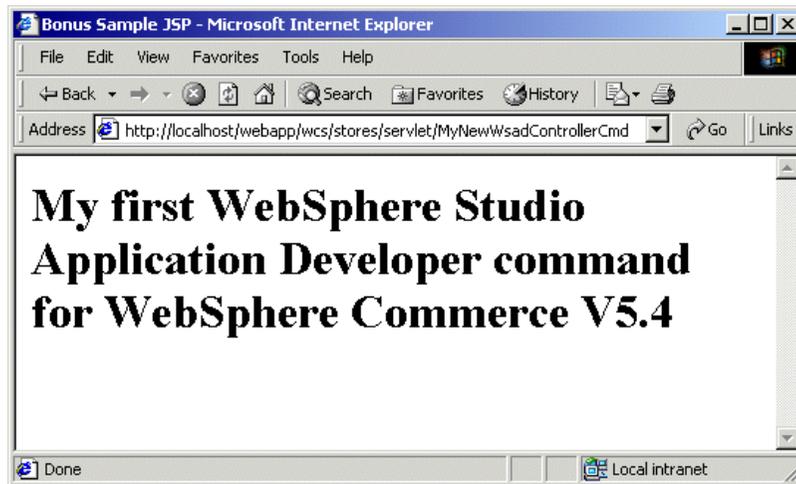


Figure 4-17 The command output



Creating a store

This chapter discusses store architecture in WebSphere Commerce Business Edition, creating new stores by either modifying the sample stores, developing new store assets or a combination of both. Details about packaging and deployment of a store are discussed in Chapter 7, “Packaging and deployment” on page 201.

5.1 Stores architecture overview

You need to have a good understanding of store architecture and the components of a store, before deciding how to create a store. The following sections provide an overview of store architecture, components and assets.

5.1.1 Store assets and components

A commerce store consist of the following assets:

- ▶ Web assets
- ▶ Business logic (Enterprise JavaBeans and commands)
- ▶ Store data

The Web assets of your store are the interface that users would be using to interact with the store, this typically consists of JSPs HTMLs and images. Users will be able to view catalog's, place or review orders through this interface

Business logic for a stores operation is created as reusable components either in the form of Enterprise JavaBeans or commands. These are often shared by all the stores in a Commerce Server. Users may customize the existing business logic to suit their business. Extending business logic and creating new business logic is extensively discussed in *IBM WebSphere Commerce Programmers Guide Version 5.4*,

Store data comprises data about catalogs,orders,shipping methods, fulfillment centers, payment options, taxes and discounts, contracts and so on. Some of this data may be shared among stores but not all data needs to be shared. Data from all the stores of a single Commerce Server instance is stored in common repository.

These assets are owned and maintained by various components of a Commerce Server. The list of components that a store would contain are

- ▶ Commerce Server
- ▶ Commerce instance
- ▶ Store configuration

WebSphere Commerce Server provides the store with related functions to run a commerce solution over the Web, the business logic and the store assets are stored in the Web application server. This provides a default Web application which can hold Web assets of a single or multiple stores. Users may also create a different Web application to hold Web assets of a single or multiple stores.

A Commerce Server instance, is an enterprise application running on the WebSphere Application Server, it may contain single or multiple stores. The instance provides the store with the required business logic to process and persist data. All the stores residing on a single commerce instance share the same Enterprise JavaBeans and commands, and they also share a common database.

5.1.2 Store design

Stores in WebSphere Commerce Business Edition follow the model-view-controller based design. In the model-view-controller based design, data is modeled in a class with has get and set methods to retrieve and set its data attributes, in this case Java beans are used as models which wrap access beans or the Enterprise JavaBeans. A controller is usually a servlet which, based on the incoming request, invokes the appropriate business process and sends a response back to the client in the form of a view. The client primarily interacts with the view.

In a commerce site, the request servlet acts as a primary controller which parses the client request and invokes the requested command using the Web controller. When the requested command is a view command the user request is forwarded/redirected to a different view. When the requested command is a controller command it invokes the appropriate task commands based on the business process and sends a response back to client.

The business logic of the commerce application is wrapped inside the task commands, controller commands invoke the appropriate task command based on the business process. Data in the database is manipulated via entity beans. Commands do not directly interact with entity beans instead they use access beans to interact. Databeans are created to model data that can be used in a view.

For all commands there are post and pre commands defined in the Commerce Server, and when you wish to modify the functionality of a command, it is best to override these pre or post commands instead of overriding the task/controller commands. For example if you wish to modify the business logic registering a user in the Commerce Server, you can either extend the `PreUserRegistrationAddCmd` or the `PostUserRegistrationAddCmd`. When you want to add functionality after the user is registered you extend the `PostUserRegistrationAddCmd`, and when you want to add or modify functionality before the user is registered in the system you can extend the `PreUserRegistrationAddCmd`.

The steps to be followed to override the command and register the command are described in detail in the *IBM WebSphere Commerce Programmer's Guide Version 5.4*.

5.2 Store development

WebSphere Commerce provides users with multiple options for creating a store

- ▶ Creating a store based on the sample stores shipped with the WebSphere Commerce.
- ▶ Creating a store by developing new store assets,
- ▶ Creating a store using a combination of the above.

Chapter Chapter 4, “Planning and development” on page 71 provides more details on the tool that can be used for creating stores.

5.2.1 Creating a store based on sample stores

WebSphere Commerce provides sample stores which showcase the capabilities of the product, and these can be used to create custom stores by modifying the store assets. The store models that are available for WebSphere Commerce Business Edition V5.4 are:

- ▶ InFashion (B2C)
- ▶ NewFashion (B2C)
- ▶ ToolTech (B2B)
- ▶ WebAuction (B2C)
- ▶ WebFashion (B2C)

Note: Not all the sample stores listed above are shipped with the product. In some cases they are available as downloads.

Each of these store models, showcase different business scenarios. For example ToolTech showcases a business-to-business scenario, InFashion and WebFashion showcase a business-to-consumer scenario. Based on their business needs a user can select the best store and customize the store.

Sample stores are shipped in a store archive (SAR) format. These file format's can be directly published to the Commerce Server using Store Services. There is also a command line option, available which can be used to publish stores on to the Commerce Server.

The following are the components of a SAR file:

- ▶ Access control policies
- ▶ Business policies
- ▶ Catalog data
- ▶ Contracts
- ▶ Commands
- ▶ Tax and shipping
- ▶ Store data
- ▶ Web assets

In addition to this the SAR file contains deployment descriptors with information about how the store should be installed.

Note: To create a store archive that can be used as a sample with Store Services, see the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*

Create organization and assign roles

An organization has to be created in the Commerce Server and assigned the role of a seller. An organization can be created from the WebSphere Commerce Administration Console. The procedure of creating an organization is:

1. Logon to Administration Console. and select **Site** in the **Site/Store Selection** window.
2. Select **Access Management -> Organizations.**
3. Click **New.** Figure 5-1 will be displayed.

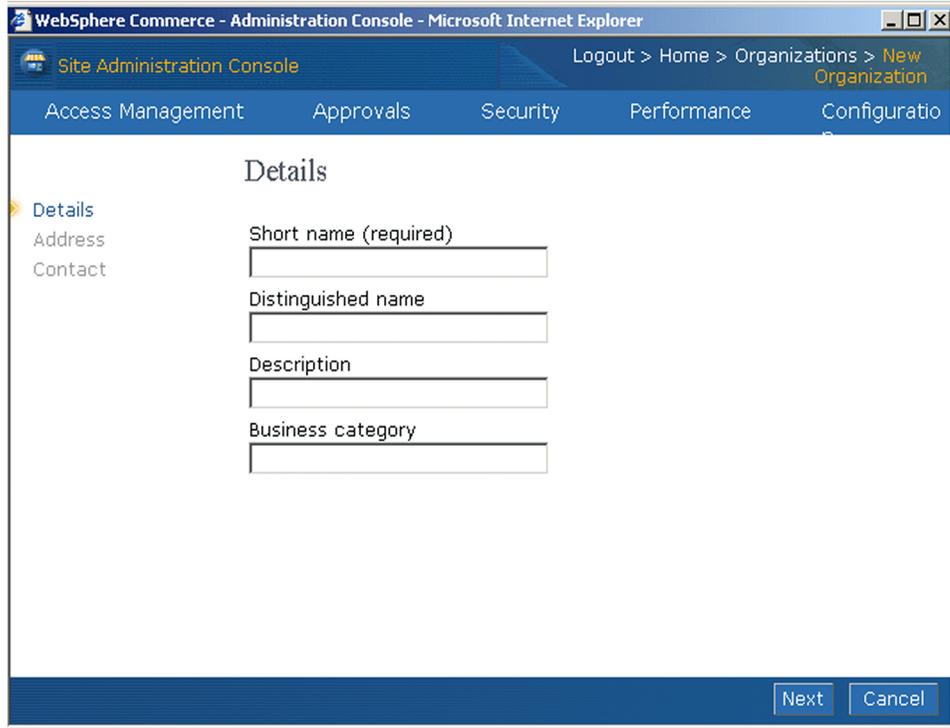


Figure 5-1 Create new organization

4. Enter Name of the organization in Short Name field.
5. Select **Organization Type**.
6. Click **Next**.
7. Provide address information.
8. Organization contact information is not mandatory.
9. Click **Finish** to complete the creation of the organization
10. Select the organization you have just created and click **Roles**.
11. Select the role which you want to assign and click **Add**.

Tip: Please make sure you add all the roles that this organization should support. The users of this organization may not have roles assigned other than the roles of the organization.

12. Click **OK** when you are done.

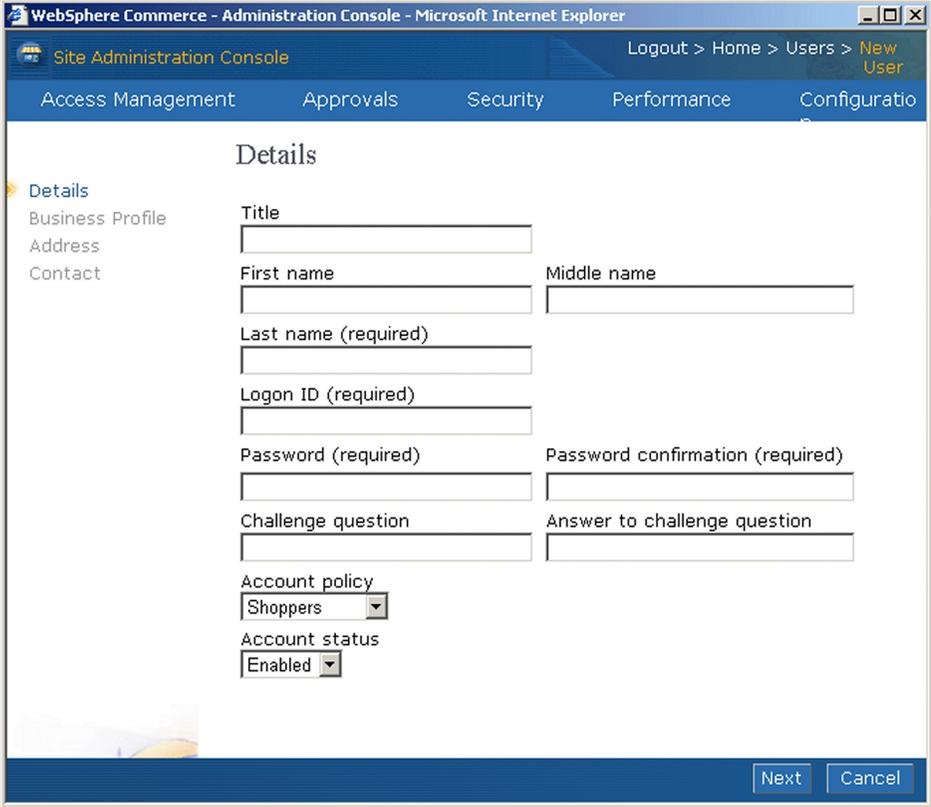
Create users and assign roles

In order to create a store in the Commerce Server, you need to have the following users created, and appropriate roles assigned.

- ▶ Site Administrator (In case you are not using the default)
- ▶ Seller Administrator
- ▶ Store Administrator
- ▶ Store Developer

What follows is list of steps to create users in the WebSphere Commerce Administration Console, and assign roles to users.

1. Log on to Administration Console and select **Site** in the **Site/Store Selection** window.
2. In the Access Management menu, select **Users**
3. Click **New**. Figure 5-2 will be displayed



The screenshot shows the 'WebSphere Commerce - Administration Console - Microsoft Internet Explorer' window. The breadcrumb trail is 'Logout > Home > Users > New User'. The navigation menu includes 'Access Management', 'Approvals', 'Security', 'Performance', and 'Configuratio'. The main content area is titled 'Details' and contains a form for creating a new user. The form fields are: Title, First name, Middle name, Last name (required), Logon ID (required), Password (required), Password confirmation (required), Challenge question, Answer to challenge question, Account policy (dropdown menu with 'Shoppers' selected), and Account status (dropdown menu with 'Enabled' selected). At the bottom right, there are 'Next' and 'Cancel' buttons.

Figure 5-2 Create new user

4. Enter data in last name, logon id, password and password verify, fields.
5. Business profiles are non-mandatory fields, if required you may enter relevant data into these fields.
6. Select the organization that the new user would belong to.
7. Enter address information
8. Enter contact information if required, it is not a mandatory.
9. Click **Finish** to complete user creation.
10. Select the user to whom you wish to assign roles, and click **Roles**
11. Select the organization that the user will play a role for, select the role, and then click **Add**.
12. Click **OK** when complete.

Change store database assets

When you select a store model to customize, the archive already has some data in the form of XML files in it. In most of the cases to modify the data you will have to edit the XML manually

Some of the store data, can be edited by using Store Services for example, you can modify shipping and tax information using Store Services, and edit the store profile. Store data like product or catalog data can be either modified directly by editing the XML before the store is published or can be modified after the store is published by using the WebSphere Commerce Accelerator. You can create new products and categories as well as associate products with categories.

Note: Details of how to create store assets are discussed in detail in *IBM WebSphere Commerce Store Developer's Guide Version 5.4*

When a store is published its data is loaded as in the order specified in the sarinfo.XML. This follows a pre-defined order, so the order of the assets you wish to create should follow the order specified in the XML. Make sure the parent table assets precede the child table assets.

Please refer the *IBM WebSphere Store Developers Guide* to get more information about creating store data assets. This manual discusses creating assets by modifying existing assets and describes the tools that are available to modify assets. It also details methods of packing assets.

Modifying Web assets

The store front which is primarily JSP, HTML, and image files is zipped and bundled as a part of the SAR file. You can extract the Webapp.zip file using Store Services or you can directly unzip it into a new folder and modify the JSPs as dictated by business needs, and then repackage the JSPs and other store front assets as Webapp.zip.

To modify Web assets, you can use WebSphere Studio, where you can import the SAR file directly to the workspace and work on the Web assets. Please refer to Chapter 4, “Planning and development” on page 71 for more information on tooling.

A standard naming convention for JSPs is maintained across sample stores, but in few cases the JSPs may have variation in the name’s used. There are few JSPs that act as controllers in each subsystem to control the navigation between JSPs.

For example the logon process is controlled by LogonForm.jsp as shown in Example 5-1.

Example 5-1 LogonForm of ToolTech Sample Store

```
//Parameters may be encrypted. Use JSPHelper to get URL parameter instead of
request.getParameter().
JSPHelper jhelper = new JSPHelper(request);

String storeId = jhelper.getParameter("storeId");
String catalogId = jhelper.getParameter("catalogId");
String languageId = jhelper.getParameter("langId");

boolean isError = false;
String[] strArrayAuth = request.getAttribute(EConstants.EC_ERROR_CODE); // (String
[]))request.getAttribute(EConstants.EC_ERROR_CODE);
if (strArrayAuth != null) isError = true;
String userType = cmdcontext.getUser().getRegisterType().trim();

String incfile = null;

if (!userType.equalsIgnoreCase("G"))
{
    // User is registered - show their User Account Profile
    incfile = storeDir + "UserAccount.jsp";
}
else
{
    // User is not registered - show Logon Page again
    incfile = storeDir + "LogonDisplay.jsp";
}
```

```
%>
```

```
<jsp:include page="<%=incfile%>" flush="true"/>
```

Based on a predefined attribute the form decides which view needs to be displayed. In the example shown above, if the customer is a guest shopper the view is redirected to the logon page, and if the customer is a registered user the view is redirected to home page.

There is a generic JSP called `GetResource.jsp` that is created in all the store models to provide NLS support. It initializes the store properties files based on the locale using information extracted from the command context. This JSP can be reused, in case you wish to provide NLS. Example 5-2 shows the `GetResource.jsp` used by the ToolTech sample store.

Example 5-2 GetResource.jsp of ToolTech store

```
CommandContext      cmdcontext      =      (CommandContext)
request.getAttribute(ECConstants.EC_COMMANDCONTEXT);
Locale locale = cmdcontext.getLocale();

String storeDir = (String) request.getAttribute("storeDir");
String fileDir = (String) request.getAttribute("fileDir");
String includeDir = (String) request.getAttribute("includeDir");
String bundleDir = (String) request.getAttribute("bundleDir");
String storeName = "";

if (storeDir == null)
{
    storeDir = sdb.getJspPath();
    fileDir = sdb.getFilePath();
    includeDir = storeDir + "include" + "/";
    bundleDir = sdb.getDirectory();
    storeName
sdb.getDescription(cmdcontext.getLanguageId()).getDisplayName();
request.setAttribute("storeName", storeName);
request.setAttribute("storeDir", storeDir);
request.setAttribute("includeDir", includeDir);
request.setAttribute("fileDir", fileDir);
request.setAttribute("bundleDir", bundleDir);
}
```

```

ResourceBundle tooltechtex = (ResourceBundle)
request.getAttribute("ResourceText");

if (tooltechtex == null)
{
    tooltechtex = ResourceBundle.getBundle(bundleDir + "/tooltechtex", locale
);
    request.setAttribute("ResourceText", tooltechtex);
}
response.setContentType(tooltechtex.getString("ENCODING"));
%>

```

You can also add new JSPs according to business needs. Make sure you have view commands created for the new JSPs and that you assign necessary access to customers so that these JSP can be used. You may also extend controller or task commands to override the business logic to suit your needs. Again care must be taken when providing access rights.

Create business accounts

Business accounts are created to establish a relation between organizations and stores in WebSphere Commerce Business Edition. Contracts are created for business accounts and they determine the cost of products, payment method, shipping methods, and approval process.

A business account has the following information:

- ▶ Customer
 - Customer organization,
 - Contact person for customer organization,
 - Whether or not shoppers can use default contract of the store.
- ▶ Representative

Department Name and Representative of the store which this customer organization is associated with.
- ▶ Purchase order

You can specify whether the customer belonging to this account can specify a purchase order number while an order is placed, and you can also restrict the customer's spending by specifying an spending limit amount for each purchase order number.
- ▶ Invoicing

If the customer receives an invoice for the order, you can specify the mode of delivery.

► **Credit line**

You can specify whether or not this account has a credit line. If it has the credit line you need to specify the account number, and billing address for the credit line

Users can create a business account by either editing the store XML or after the store is published, they can create a new business account using the WebSphere Commerce Accelerator

The steps to create a business account for a given customer organization are as follows:

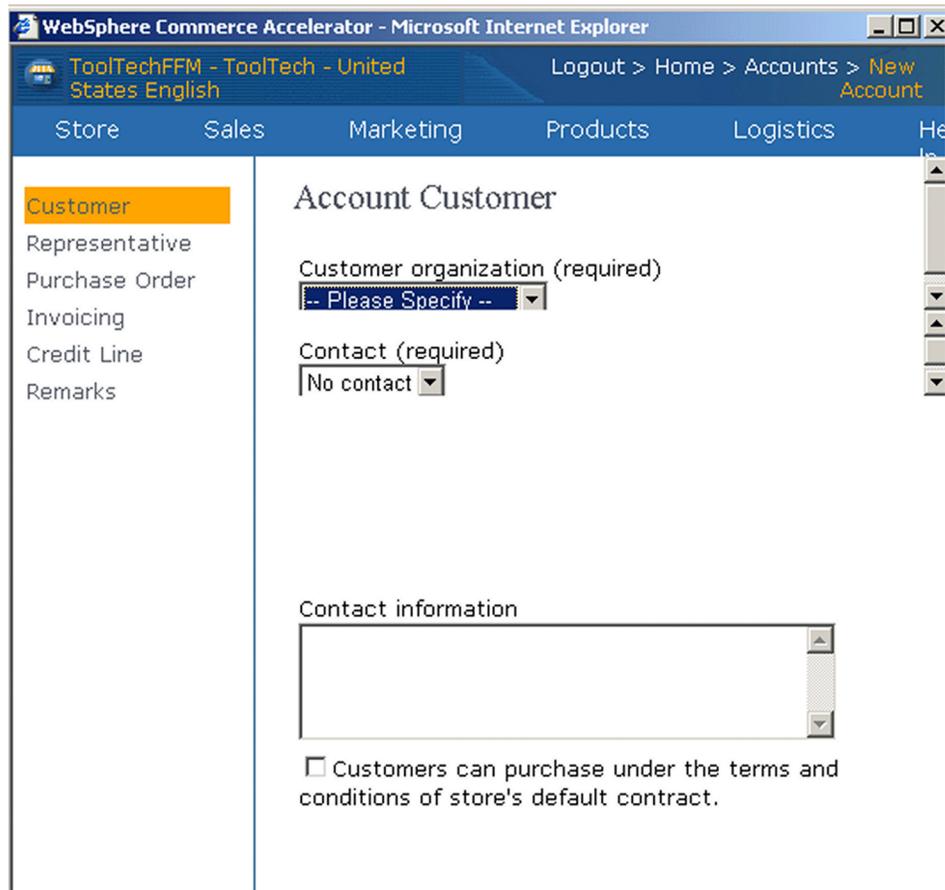
1. Log on to WebSphere Commerce Accelerator and select **Sales -> Accounts** as shown in Figure 5-3.



Figure 5-3 WebSphere Commerce Accelerator home page

2. Click **New** to create a new business account.

3. Enter customer organization information as shown in Figure 5-4. In order to have a contact person for the customer organization this business account belongs to, first create a user and assign necessary roles to the user.



The screenshot shows a web browser window titled "WebSphere Commerce Accelerator - Microsoft Internet Explorer". The address bar shows "ToolTechFFM - ToolTech - United States English" and the breadcrumb trail is "Logout > Home > Accounts > New Account". The navigation menu includes "Store", "Sales", "Marketing", "Products", "Logistics", and "Help". The left sidebar has a menu with "Customer" selected, and other options: "Representative", "Purchase Order", "Invoicing", "Credit Line", and "Remarks". The main content area is titled "Account Customer" and contains the following fields:

- Customer organization (required)**: A dropdown menu with the selected value "-- Please Specify --".
- Contact (required)**: A dropdown menu with the selected value "No contact".
- Contact information**: A large empty text area.
- Customers can purchase under the terms and conditions of store's default contract.

Figure 5-4 Enter customer organization information

4. You may also specify whether you want to allow users from this customer organization to shop using the default contract of the store.
5. Select the account representative of the store and the department of the store as shown in Figure 5-5

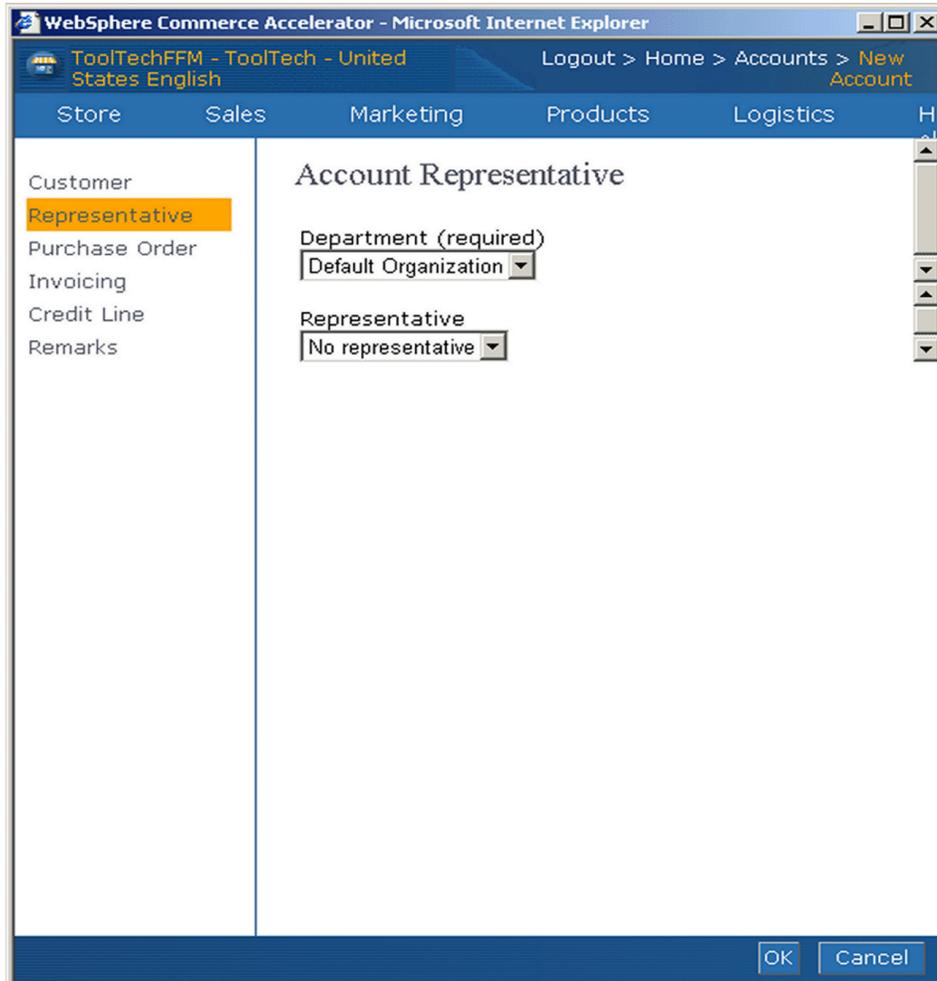


Figure 5-5 Account representative

6. You may specify a purchase order number for this customer organization that users need to enter while shopping. You may also give a constraint over the order value if the purchase order number is used. You can have different purchase order number created for different order values. See Figure 5-6 for an example.

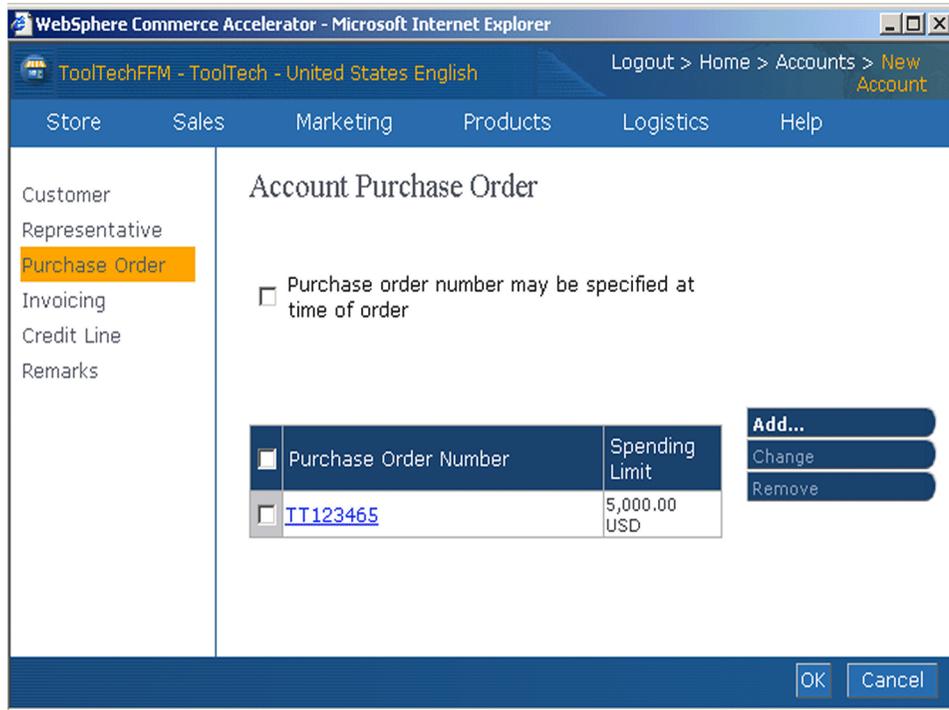


Figure 5-6 Purchase order

7. If the store provides an invoice to the customer organization for every order placed, then the business account should specify the mode of delivery. By default there are three delivery modes available. These are shown in Figure 5-7.

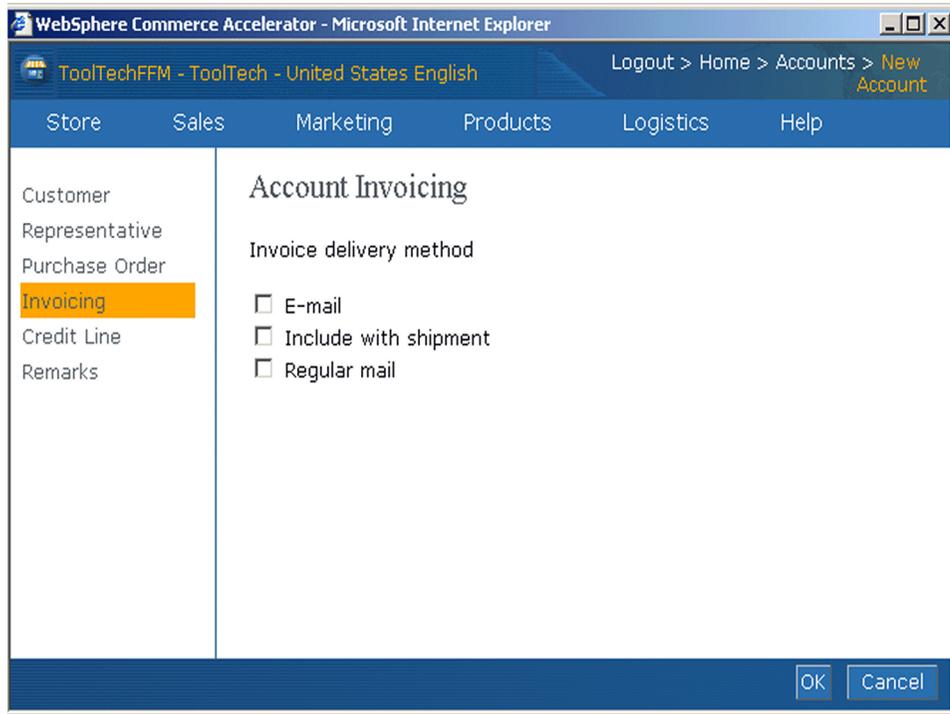


Figure 5-7 Invoicing

8. If the customer organization has a credit line, you can specify the credit line number and the billing address. If you do not define a credit line, the contract associated with this business account cannot have a payment option of credit line. Other payment methods will have to be assigned to the contract. See Figure 5-8 for an example.

ToolTechFFM - ToolTech - United States English Logout > Home > Accounts > New Account

Store Sales Marketing Products Logistics Help

Customer
Representative
Purchase Order
Invoicing
Credit Line
Remarks

Account Credit Line

The account has a credit line

Description (required)
Buyer org D credit line

Credit line account number
ORGD123456

Billing address
No address specified

OK Cancel

Figure 5-8 Credit line

9. Enter remarks if you have any and click **OK**.

Once the business account has been created for a customer organization, you will have to create contracts for this business account. Contracts can be created by either modifying the XML file in the SAR or you can create a contracts using the WebSphere Commerce Accelerator.

Create contract

Contracts enable customers associated with a particular business account to buy products from a commerce site, for a specified price, under pre-defined conditions. A business account can have one or more contracts, and each contract may offer the same product at different prices and use different shipping and payment methods.

Contracts are specific to a business account, a customer organization cannot have any contracts without an business account.

A contract in a commerce site offers the following:

- ▶ Every contract in a Commerce Server has a unique identification and a period for which the contract will be effective.
- ▶ It lists customer organizations that this contact will be applicable for and only users from these customer organizations will be able to use the contract. A contract always involves both a buyer and a seller organization.
- ▶ Terms and conditions, that actually determine what the price of a product or item would be, and which also provide information about returns, refunds, payment, shipping modes, and order approval.
- ▶ The contract has pointers to location where a detailed description of this contract is available.

Contracts can be created when the store is published or after the store is published. If you wish to create the contact before the store is published, extract the contract.XML from the sample store that you wish to customize, and edit the contract manually. To create a contract after the store is published use WebSphere Commerce Accelerator. Please make sure you have a business account created, before creating a contract for a customer organization.

The steps to create a contract using the WebSphere Commerce Accelerator are as follows:

1. Log on to WebSphere Commerce Accelerator and select the commerce site that you wish to work on, then select **Sales-> Accounts**. The page that is displayed will have the list of business accounts that exist for this commerce site. If you want to add a contract to a existing account, you just have to select the business account and click **New Contract**, if you wish to list the available contracts for a business account, select the business account and click **Contracts**. Creating a business account is discussed in “Create business accounts” on page 141.

Figure 5-9 shows the list of contracts in a sample store.

WebSphere Commerce Accelerator - Microsoft Internet Explorer

ToolTechFFM - ToolTech - United States English Logout > Home > Accounts > Contracts

Store Sales Marketing Products Logistics Help

Contracts

View: Page Number:

2 items << [First](#) | 1 of 1 | [Last](#) >>

	Name	Short Description	Start	End	Status	Created	
<input type="checkbox"/>	ToolTech Contract #6	ToolTech Contract #6	5/7/02 12:00 AM	No expiry date	Active	5/7/02 12:31 PM	New... New Version Change Submit Deploy Resume Suspend Summary Reports Duplicate... Cancel Delete
<input type="checkbox"/>	ToolTech Contract number 2345	Contract 2345		No expiry date	Active	4/19/02 5:57 PM	

Figure 5-9 Contracts

- Enter the contract general information like name, short description. You may also set an expiry date for this contract, by unchecking the no expiry date check box, and entering an expiry date. You may also set a date for the contract to be effective. See Figure 5-10 for an example.

ToolTechFFM - ToolTech - United States English Logout > Home > Accounts > New Contract

Store Sales Marketing Products Logistics Help

General

- Customers
- Products and Prices
- Selection Constraints
- Shipping
- Payment
- Returns
- Order Approval
- Attachments
- Remarks

Contract General Information

Contract name (required)

Short description (required)

Description

Start date (required) Year: 2002 Month: 05 Day: 09 Time: 00:00

No expiry date

End date (required) Year: 2003 Month: 05 Day: 09 Time: 00:00

Figure 5-10 Contract profile

3. You can specify a price for a product in two ways, you can either specify percentage pricing or a fixed pricing.

In percentage pricing you can either markup or markdown the price of products in the entire catalog by some percentage. You can also limit the price change to some specific categories or products.

Using fixed pricing, you can specify a fixed price for products that will be purchased under this contract.

See Figure 5-11 for an example.

The screenshot shows a web browser window titled "WebSphere Commerce Accelerator - Microsoft Internet Explorer". The browser's address bar shows "ToolTechFFM - ToolTech - United States English". The breadcrumb navigation path is "Logout > Home > Accounts > New Contract > Add Percentage Pricing". A blue navigation bar contains links for "Store", "Sales", "Marketing", "Products", "Logistics", and "Help".

The main content area is titled "Add Percentage Pricing" and contains two radio button options:

- Apply an adjustment on the entire master catalog
 - Adjustment (required)
 - %
- Apply an adjustment on the following categories and products

At the bottom right of the form are "OK" and "Cancel" buttons.

Figure 5-11 Pricing Information

4. You can specify the shipping providers that this contract supports and also provide the charge type, and the shipping address as shown in Figure 5-12. The default shipping address would be the address of the customer organization.

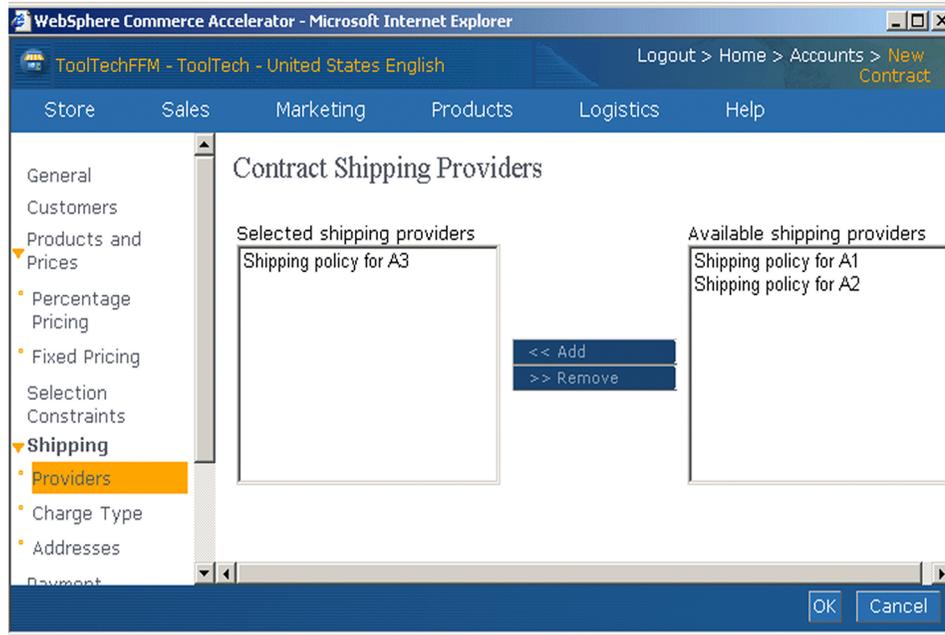


Figure 5-12 Shipping details

5. You can also specify the payment methods that this contract supports. If the business has a credit line then you can select credit line as a method of payment. The other payment options are picked from Payment Manager. To install and configure Payment Manager refer to *IBM WebSphere Payment Manager for Multiplications Installation Guide*.
6. Click **Add** to add payment methods as shown in Figure 5-13.

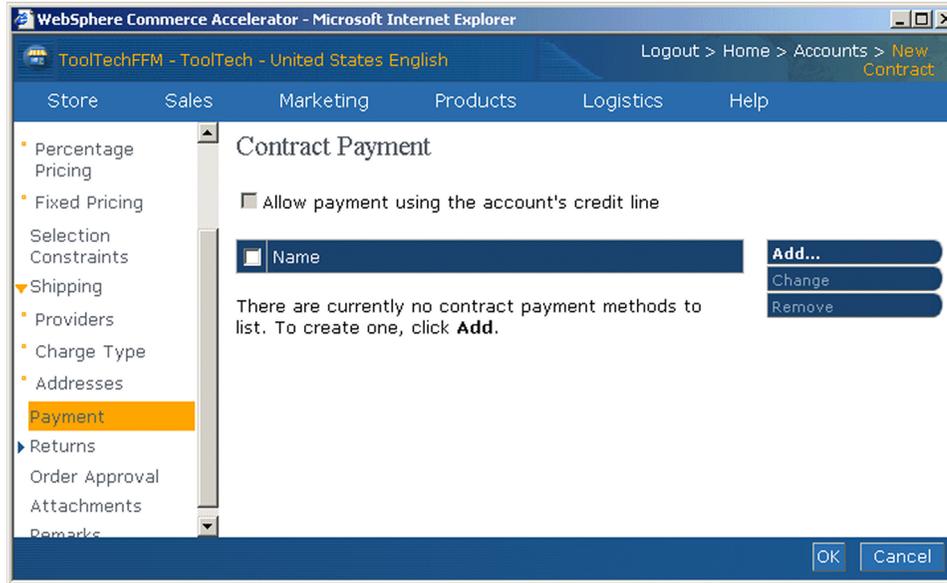


Figure 5-13 Payment methods

7. To specify the return and refund policy, click **Return** and select the return policy. You may also specify refund terms and conditions. A refund can be made through a credit line or through the other payment methods.
8. You can provide pointers to a location where details about this contract are available.
9. Once the contract is complete click **OK**.
10. To deploy a contract, select the contract that you have created and, click **Submit**.
11. Once the contract is successfully deployed the status should read active.
You do not have to restart the commerce instance for this contract to be in use.

Tip: To refresh the registries of Commerce Server do not restart the instance but, logon to the Administration Console, and select **Configuration -> Registry**. Select the registries you want to refresh and click **Refresh**. If you wish to refresh all the registries click **Update All**

Access control policies

The earlier versions of WebSphere Commerce provided coarse grained access control, where you could assign permissions for a user group. In contrast WebSphere Commerce Business Edition V5.4 provides fine grained access control. You can define which user groups can invoke which functions of a given business object. For example, buyers can be given access to delete orders of their own, and not the orders of other buyers at the same customer organization. This access is provided at the instance level as opposed to the method level.

Access control policies are the rules that which determine users belonging to a user group can perform actions in a commerce site. You can provide access to all operations that can be performed on a commerce site. Access control is now externalized in WebSphere Commerce Business Edition V5.4, so you may customize access control policies without making any modifications to the code, all that you will have to do, is modify XML files.

Access control in WebSphere Commerce Business Edition, has the following components:

- ▶ Access group
A group of users who share common access to a set of business object.
- ▶ Action group
A set of actions that can be performed on business objects.
- ▶ Resource group
A set of resources that can be controlled by a common policy. They can be a group of commands, views, or contracts and so on.
- ▶ Relationship
In some cases you may have to create a relationship between resource groups and user groups so that only user groups who are related to the resource group can perform actions on it. For example a user who owns a business object in commerce site is authorized to modify it, other users belonging to the same access group, and having common rights on the resource group may not be able to modify the business object.

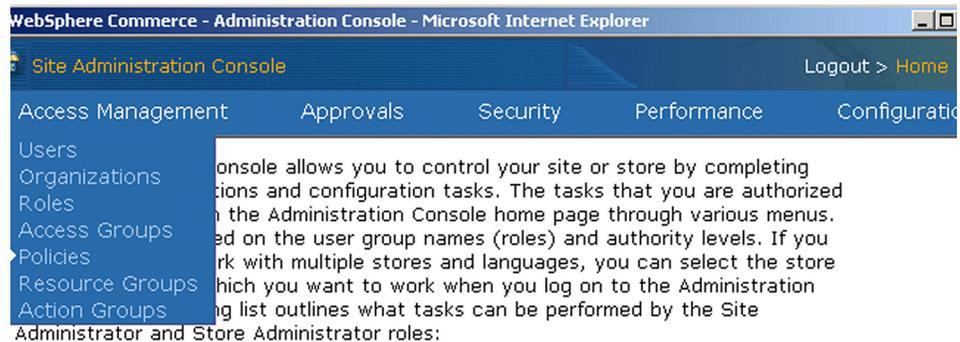
Later in this chapter we discuss how to create access policies for the commands or views that you have created for a commerce site.

Note: For details about access control policies, refer to the *IBM WebSphere Commerce Access Control Guide Version 5.4*

To create access policies for your customized store, you can either use the existing user groups or create new user groups and provide access to resources. You can create policies by either editing the XML file, or by using the Administration Console.

Create access group

1. Log on to Administration Console and select **Access Management** -> **Access Groups** as shown in Figure 5-14.



Site Administrator

- Manages users, organizations, roles, access groups, policies, resource groups and action groups.
- Monitors performance
- Configures messages
- Manages payments and specifies settings
- Approvals, including user registration, orders, RFQ responses, and contract summaries.
- Configures logging and tracing
- Enables and disables WebSphere Commerce components
- Changes security policies

Store Administrator

- Configures messages
- Administers Blaze rules
- Manages payments and specifies settings

Figure 5-14 Administration Console access management

2. A list of access groups that are currently active in the Commerce Server instance is displayed as shown in Figure 5-15. You may either customize existing access groups or create new access groups. Click on **New** to create new access group.

WebSphere Commerce - Administration Console - Microsoft Internet Explorer

Site Administration Console Logout > Home > Access Groups

Access Management Approvals Security Performance Configuration

Access Groups

Page Number Go

115 items << First 1 of 6 Next >> Last >>

<input type="checkbox"/>	Name of access groups	Description of access groups
<input type="checkbox"/>	AccountAdministratorsForOrg	Users who can manage accounts for the organization
<input type="checkbox"/>	AccountHandlersForOrg	Users who can handle accounts in the organization
<input type="checkbox"/>	AccountManagersForOrg	Users who can manage accounts in the organization
<input type="checkbox"/>	AccountRepresentatives	Users with role of account representative
<input type="checkbox"/>	AccountRepresentativesForOrg	Users with role of account representative within the organization
<input type="checkbox"/>	AccountViewersForOrg	Users who can view account details in the organization
<input type="checkbox"/>	Administrators	Administrators
<input type="checkbox"/>	AllUsers	All users
<input type="checkbox"/>	AuctionAdministratorsForOrg	Administrators of auctions in the organization
<input type="checkbox"/>	AuctionManagersForOrg	Users who can manage auctions
<input type="checkbox"/>	B2BCSAViewUsersForOrg	Users with role of B2BCSA view users for the organization
<input type="checkbox"/>	B2CCSAViewUsersForOrg	Users with role of B2CCSA view users for the organization
<input type="checkbox"/>	BackendOrderAdministratorsForOrg	Administrators of backend orders for the organization

New

Change

Delete

Show Actions

Show Resources

Show Policies

Figure 5-15 Access group list

- Enter the name of the access group you wish to create in the name field. You may also enter a description for this access group. Select the organization this access group will belong to and click on **Next**. See Figure 5-16.

Details

Details
Criteria

Name (required)

Description

Parent Organization

- Root Organization
 - Default Organization
 - Seller Organization
 - Buyer Organization A
 - Buyer Organization B
 - Seller Organization A
 - Buyer Organization C
 - Buyer Organization D



Figure 5-16 Access group profile

4. You can have two kinds of access groups, implicit and explicit access groups. An implicit access group is defined by a set of conditions, anyone who complies to these set of conditions becomes a member of this access group. You can either have simple implicit conditions like, members with a specific role regardless of organization can be a part of this access group, or you may create complex rules, like members can be a part of this group, if they have a specific role in a specific organization.

You may also include members based on their registration status, all members having a common registration status will become a part of the access group.

When you select no criteria you will include all users of the commerce site in this access group.

Figure 5-17 shows an example of access group criteria.

Criteria

[Details](#)
[Criteria](#)

No criteria
 Based on organizations and roles

Organization

Role

For organization Add

Selected roles and organizations
 Remove

Based on registration status



Figure 5-17 Access group criteria

5. Click **Finish** to complete creation of the access group.

Create action group

Action groups are a collection of actions that can be performed by an access group in a commerce site. You can create new actions if required or use existing actions, and later create action groups. The action names of view commands are the same as the view name or command name. Since it execute action that is performed on controller and task commands the action name for controller commands is `executecommand`, and the command name is `execute`. For actions pertaining to databeans the action name is `displaydatabean` and the command name is `displayed`. Example 5-3 shows XML used to define actions.

Example 5-3 Action

```

<!-- Executre a View Command Resource-->
<Action Name="AddFeatureDialogView"
      CommandName="AddFeatureDialogView">
</Action>
    
```

```

<Action Name="PASummaryDialogView"
        CommandName="PASummaryDialogView">
</Action>

<Action Name="ProductComparerListView"
        CommandName="ProductComparerListView">
</Action>

<!-- Execute a controller command resource -->
<Action Name="ExecuteCommand"
        CommandName="Execute">
</Action>

<!-- Allows 'display' on databean resources -->
<Action Name="DisplayDatabean"
        CommandName="Display">
</Action>

```

Example 5-4 shows the XML that defines action groups.

Example 5-4 Action group

```

<!-- Action group enabling execution of command-resources -->

<!-- Action Group for Controller Commands -->
<ActionGroup Name="ExecuteCommandActionGroup"
        OwnerID="RootOrganization">
    <ActionGroupAction Name="ExecuteCommand"/>
</ActionGroup>

<!-- Allows 'display' on databean resources -->
<ActionGroup Name="DisplayDatabeanActionGroup"
        OwnerID="RootOrganization">
    <ActionGroupAction Name="DisplayDatabean"/>
</ActionGroup>

<!-- Action Group for View Commands -->

<ActionGroup Name="PAWCBEViewsActionGroup"
        OwnerID="RootOrganization">
    <ActionGroupAction Name="PACategoryListView"/>
    <ActionGroupAction Name="AddPEFeatureDialogView"/>
    <ActionGroupAction Name="PASummaryDialogView"/>
    <ActionGroupAction Name="ProductComparerListView"/>
    <ActionGroupAction Name="ProductExplorerListView"/>
    <ActionGroupAction Name="PADynamicListView"/>
    <ActionGroupAction Name="PADialogNavigation"/>

```

```
</ActionGroup>
```

Once you define the list of actions that a member group can perform in the commerce site, you can logically group a set of actions into an action group. Every action group has a owner associated with it, the owner of the action group is the parent organization as shown in Example 5-4.

Create resource group

A resource category is a collection of resources that need to be protected by access control. You can associate existing resource categories to a resource group through the Administration Console, or by using XML. If you wish to add new resource categories you must use XML.

Resource categories can be any assets of the Commerce Server including commands, auctions , members and so on. Each resource group has a owner associated with it and the owner is usually the parent organization.

There are two kinds of resource groups, one is an implicit resource group and the other is explicit resource group. Implicit resource groups define a set of resources that have common attributes and values. For example for a given store you may give access to all the items in the catalog. You can also create explicit resource groups by creating resource categories and explicitly assigning resource categories to resource groups. You have to define resource actions for resource categories. These are the actions that can be performed on this resource category. For view commands actions need not be defined, but for databeans and controller commands you will have to specify actions. See Example 5-5 for a sample XML definition of a resource category.

Example 5-5 Resource category

```
<ResourceCategory Name="com.ibm.commerce.command.ViewCommandResourceCategory"
  ResourceBeanClass="com.ibm.commerce.command.ViewCommand">
  </ResourceCategory>

  <ResourceCategory
Name="com.ibm.commerce.pa.admin.builder.MetaphorFeatureDataBean"
ResourceBeanClass="com.ibm.commerce.pa.admin.builder.MetaphorFeatureDataBean">
  <ResourceAction Name="DisplayDataBean"/>
  </ResourceCategory>

  <ResourceCategory
Name="com.ibm.commerce.pa.admin.builder.CreateMetaphorControllerCmd"
ResourceBeanClass="com.ibm.commerce.pa.admin.builder.CreateMetaphorControllerCmd">
```

```
        <ResourceAction Name="ExecuteCommand" />
</ResourceCategory>
```

You can group resource categories that fall under a common classification into a resource group. All resources have a owner associated with them and other relationships can exist. See Example 5-6 for a sample XML definition of a resource group

Example 5-6 Resource group

```
<!-- View Command Resource Group -->

<ResourceGroup Name="ViewCommandResourceGroup" OwnerID="RootOrganization">
    <ResourceGroupResource
Name="com.ibm.commerce.command.ViewCommandResourceCategory" />
    </ResourceGroup>

<!-- DataBean Resource Group -->

    <ResourceGroup Name="PAWCBEDataBeanResourceGroup"
OwnerID="RootOrganization">

        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.MetaphorFeatureDataBean" />
        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.MetaphorFeatureListDataBean" />
        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.PACategoryDataBean" />
        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.PACategoryListDataBean" />
        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.PASummaryDataBean" />
        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.WidgetListDataBean" />
        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.DefaultFeatureListDataBean" />

    </ResourceGroup>

<!-- Controller Command Resource Group -->

    <ResourceGroup Name="PAWCBECommandResourceGroup"
OwnerID="RootOrganization">

        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.DeleteMetaphorControllerCmd" />
        <ResourceGroupResource
Name="com.ibm.commerce.pa.admin.builder.CreateMetaphorControllerCmd" />
```

</ResourceGroup>

Create policy

A policy associates access groups with resource groups and specifies the actions that can be performed by the access group on the resource. There are two kinds of policies in Commerce Server,

- ▶ Standard pPolicy

A standard policy is usually owned by an organization and all members directly descended from this organization will get access to resources. If the owner of a policy is the root organization, this policy is applicable site wide.

- ▶ Template policy

Template policies do not have any fixed owners, the owners are dynamic. This policy applies dynamically to the an organization unit which owns the resource and to its parent organization entity. You can create policies for a commerce site either by using Administration Console or by manually modifying the XML file.

To create a policy using the Administration Console:

1. Log on to the Administration Console and select **Access Management -> Policies**.
2. Click **New** Figure 5-18 will be displayed
3. Enter a unique name for your policy in the name field. You may also enter an displayname and a description for the policy as shown in Figure 5-18.

Access Management Approvals Security Performance

New Policy

Name
ToolTech Users

Display Name
ToolTech Users Policy

Description
ToolTech Users

User Group
AllUsers Find...

Resource Group
CampaignDatabaseResourceGroup

Action Group
DisplayDatabaseActionGroup

Relationship
none

Policy Type
 Template Policy

OK Cancel

Figure 5-18 New policy

4. To add the user group who will be governed by this policy, click **Find**. From the list of users groups that are available in the Commerce Server, you can select one user group per policy. Select the user group you wish to add and click **OK**. See Figure 5-19 for an example.

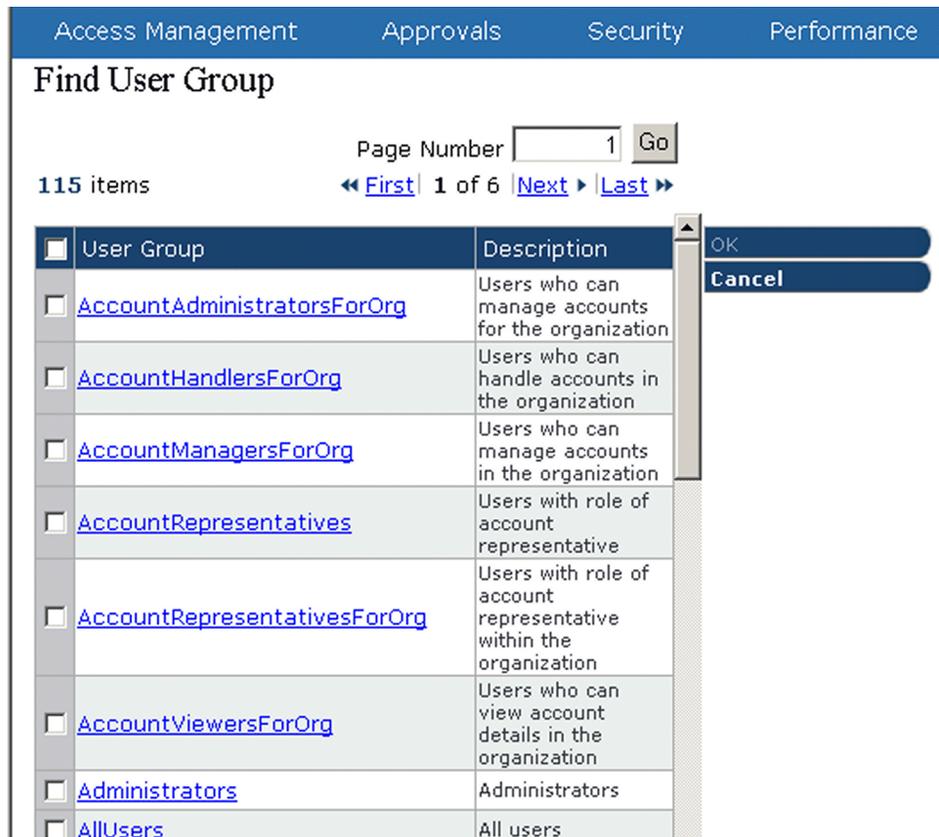


Figure 5-19 Find user group

5. Select the resource group to be accessed, from the drop down list of available resource groups.

Note: If you wish to create more resource groups use the Administration Console and select **Access Management -> Resource Groups**.

6. Select the action group from the drop down list of available groups.

Note: If you wish to create more action groups use the Administration Console and select **Access Management -> Action Groups**. New actions cannot be created using the Administration Console. Instead you will have to manually create or edit an XML file and upload the data into the Commerce Server

7. Select the type of policy you wish to create and click **OK** when complete.

Upload access policy using a command line

Not all policy assets can be created using the graphical interface tools in the Administration Console. For example, actions and resources cannot be created using the interface, instead you may have to create a XML file and upload data into the WebSphere Commerce database. Examples of how to create these XML files were given in “Create action group” on page 158 and in “Create resource group” on page 160. In this section we look at the methods of mass importing the data from these XML files into the WebSphere Commerce database using command line utilities. Command line utilities are provided to load the following data to commerce repository:

► ACUGLOAD

This utility is used to load User Groups to the commerce repository, You can create a new XML file or modify an existing user group XML file. Example 5-7 shows how to use ACUGLOAD.

Example 5-7 ACUGLOAD

```
<commerce dir>\bin\acugload <database name> <database user> <password>  
<input XML filename>
```

Sample Usage

```
C:\WebSphere\CommerceServer\bin\acugload      mall      db2admin      db2admin  
ACUserGroups_en_US.XML
```

► ACPLOAD

This utility is used to import access control policies to the repository. As well as importing access control policies, you can also use this utility to import actions, action groups, resources and, resource groups. Example 5-8 shows how to use ACPLOAD

Example 5-8 Policy load (ACPLOAD)

```
<commerce dir>\bin\acpload <database name> <database user> <password>  
<input XML filename>
```

► ACPNLSLOAD

This utility is same as ACPLOAD, except that it also provides national language support.

5.2.2 Creating store by generating new assets.

In 5.2.1, “Creating a store based on sample stores” on page 134 we discussed how to create stores based on a store model. In this section we discuss creating stores from scratch. The assets that need to be generated for creating an online store in WebSphere Commerce are.

- ▶ Web assets (This include’s JSPs, HTMLs, images.)
- ▶ Store functional data assets
 - Access control assets.
 - Business assets. (Business account, Business policy)
 - Catalog assets.
 - Contract assets.
 - Taxes, shipping and fulfillment assets.
- ▶ Store configuration data assets.

Web assets

To create Web assets you may use WebSphere Studio or WebSphere Studio Application Developer. By using WebSphere Studio you can publish the files to the server and test the files on the server. WebSphere Studio interoperates with VisualAge for Java. You can create a new servlet in VisualAge for Java and import it into WebSphere Studio. You can either follow the standard naming conventions that the other store models use for JSPs of specific subsystems or use different names. To control the page-to-page flow of JSPs create JSPs which include or append different JSPs based on a flag or a predefined value for an attribute. You can also create a view command for every single JSP in the store and control the page-to-page flow by using these view commands. In this case a single JSP will not be used to display different content

.JSPs access data from the database via databeans or through access beans. If required you can extend the WebSphere Commerce databeasn or access beans to suite business needs. Controller and task commands can also be extended. If you wish to add new fields to the database you can extend the tables as long as you take care to maintain the referential integrity of data. If you want to customize entity beans do so by extending the entity bean. We do not recommend directly changing existing entity beans.

To provide national language support(NLS), the store models implement this feature in a single JSP which loads a property file based on the locale. This JSP is used by rest of the JSPsin the store, to get the value of a property related to the locale. By providing the national language support in individual JSP, we ensure that all the other JSPs will not have to load the property file so that they

will not have to keep track of any changes of locale. The Web assets for a new store are packaged into a webapp.zip file and the properties files are packaged into a properties.zip file. After the store is published these files are extracted to the store directory under the wcsstores.war folder.

Store functional data assets

The data assets that needs to be generated for a new store are listed in “Store functional data assets” on page 166. There are some tools available to modify these data assets, but in some cases you will have to edit XML files to make modifications to the data. For example, to modify tax and shipping assets you can use the tools provided by Store Services. For modifying other assets like the catalog you have to edit an XML file, before the store is published. If you want to modify this asset after the store is published you can use the tools provided by WebSphere Commerce Accelerator. Most of the data assets are language specific, you need to generate assets to support all the languages the store supports. The steps to create each of these assets are discussed in detail in the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*.

Store configuration data assets

These are that list of assets which define configuration parameters of a store such as the default language supported by the store, the fulfillment centers, store type and so on. When data is loaded into the Commerce Server repository it is loaded in the order specified in the sarinfo.xml file which is part of the SAR file. This file acts as a descriptor for the store archive file. The data should be loaded such that the parent table is populated before populating the child tables. For more information on the order in which the data needs to be loaded refer to file <WCS_DIR>\xml\sar\sarinfo.dtd

The file store-catalog.xml specifies the catalogs and catalog entries associated with the store. You can have multiple catalogs hosted in a Commerce Server and share the catalogs and catalog entities amongst stores. This store-catalog.xml file also allows specification of category groups associated to a store. Store default values are specified in the store-defaults.xml file. defaults include the default shipping mode for the store . In order to calculate shipping charges by weight of a product, the weight of each catalog entry has to be mentioned in the CATENTSHIP table. To import this data you will have to create a store-catalog-shipping.xml file. All catalog entries in a store will need an entry in this file if shipping by weight needs to be enabled for your store. There are no specific tools available to create these XML files, you can use any of the XML editors that are currently available or use WebSphere Studio to create these XML files.

5.2.3 Create store using a mixed approach.

In 5.2.1, “Creating a store based on sample stores” on page 134 and in 5.2.2, “Creating store by generating new assets.” on page 166 we discussed how to create a commerce site by customizing a sample store or by developing new store assets. Each of these approaches has different advantages, but when a customer wants to build store and the required assets are similar to the assets of the store models it is easier to customize an existing sample store rather than develop all new store assets.

If the required store data assets of the proposed commerce site are similar to those of the sample stores, but the page-to-page flow of the new store needs to be changed then we suggest you reuse the sample store data, and then either modify the Web assets of the sample store or create new Web assets. After that you should package these assets to create a new store. There is mandatory data that needs to be created for a store and this data can be reused from the store models. Modifying the existing XML data file requires less effort than creating new files.

To reduce the development effort associated with creating Web assets you can reuse the code available in a store model. For example, to create an order approval page, you might want to modify the look and feel of the ToolTech order approval page and reuse the code. Each store model has some unique features so you can consider reusing code from all the different models if it meets your needs. For example, if you have installed WebAuction model which showcases the auctions features of you could reuse the code that is used to display products or items in auctions.

This mixed approach of creating a store works best when you want to create a store which has functions similar to those of an existing store model. You can make changes in the look and feel of the model or alter the page to page flow of the model, and still keep the existing functions intact. In most of the cases store catalog data will not be the same as that in the store model, but you can still use the catalogs of the store models as a base and modify to catalog to suite your business needs.



Testing a store

This chapter explains how to define a testing strategy for e-commerce applications, and provides guidelines on how to perform complete and exhaustive tests before deploying a store.

It includes the following sections:

- ▶ Testing strategy, containing a summary of what to keep in mind when defining the testing objectives for e-commerce applications
- ▶ Test planning, describing the test phase administrative planning and explaining the problem management and version control processes
- ▶ Test phases, containing an overview of the test process phases and a description of the typical types of testing for each phase, with some suggestions about useful test tools
- ▶ Test cases design, explaining how to design test cases and to generate results reports
- ▶ Test environment set-up, describing how to set-up staging servers and test data
- ▶ Problem determination, explaining how to use the logging and tracing features in order to identify and solve problems, with some code examples.

Note that the code reviews for checking compliance to coding standards and best practices are considered as part of the very first test phase.

6.1 Testing strategy

Testing is a fundamental process in developing high quality applications. Before shipping an application, you need to test it in order to find out whether it meets the original requirements and to verify its functionality and performance. Given the fact that an exhaustive test is virtually impossible and economically not feasible, it is crucial to identify some clear objectives and to derive from them a testing strategy that optimizes resources and reduces development cycle time.

Test objectives are the focus areas that must be tested in order to assure that applications goals are achieved, and they are based on business functions, technical requirements and project risks. Testing strategy is a high-level specification of the testing tasks that must be accomplished in order to achieve such objectives.

Typical focus areas for e-commerce applications are the following:

- ▶ Usability. Easy to navigate stores can capture the attention of a wider range of potential Web customers, including those Web navigators who are new to e-commerce and online stores.
- ▶ Scalability and performance. The customer's experience of an online store will be crucial in determining their disposition to actually purchase the displayed products. Any loading page time-out or failure will be quickly discovered by potential customers, making them move to competitors sites.
- ▶ Security. All the online store members, such as service provider, customers and business partners, must be sure that confidential information provided during monetary transactions are protected against loss, corruption and misuse by either deliberate or accidental actions.
- ▶ Systems integration. E-commerce applications are usually the result of the cooperation and integration between several applications, residing on different systems and implemented with different technologies.
- ▶ Short development cycles. Time to market is a crucial factor for an e-commerce application and this often means short development cycles, usually resulting in badly tested code with a low level of quality.

A possible testing strategy based on the focus areas, project risks, and exposures detailed above, is the following:

- ▶ Testing usability must be an important part of the system test phase.
- ▶ Scalability and performance criteria has to be established at the application design phase (see also 2.3.2, "Topology configuration samples" on page 31). Adequate measurements must begin during the functional test phase and continue for all the system test phase. The following sections of this chapter contain suggestions about tools that monitor and stimulate Web sites.

- ▶ Attention to security issues must begin during the application design phase, and security testing has to be a considerable part of the system test efforts. Again, the following sections of this chapter contain suggestions about security testing tools.
- ▶ The system test environment must have an hardware, software and network configuration as similar as possible to the production environment. Moreover, test cases must cover all the functionality involving system integration.
- ▶ Project risks and exposures derived by a short development cycle must be addressed and minimized by finding out and setting up the correct level of automation for the testing process, as well as by using test tools. The following sections of this chapter contains some guidelines about test automation and test tools.

6.2 Test planning

Test planning defines the process schedule, the team organization and the resources required to set up the test environment, as well as the processes of problem management and version control.

6.2.1 The test administrative plan

Developing a test administrative plan starts with the identification and sizing of test phases according to the defined testing strategy (see the following 6.3, “Test phases” on page 173 for details). For a realistic sizing, the testers skill and experience with e-commerce applications and products must be taken into consideration.

Moreover, for each test phase, entry and exit criteria must be defined, and specific schedules must be developed according to such criteria and to the functions delivery plan. Even if they are often affected by changes during the development phase, test schedules play a key role in identifying and controlling project risks and exposures.

A complete test administrative plan deals also with the test team organization, assigning roles and responsibilities inside the team, and with hardware and software requirements for test tools and for all the resources necessary to set up a testing environment as similar as possible to the production environment.

6.2.2 Problem management

Problem management is the process by which problems, that is application defects, are identified, tracked and kept under control in a development project.

The following is a list of all the possible statuses that a defect assumes during the test phase:

- ▶ **Open.** A defect has been discovered during one or more test executions and a description of the defect and of the circumstances on which it has occurred is provided for the development team. Usually, a severity is assigned in order to prioritize defect fixing.
- ▶ **Working.** The development team accepts the defect as a recognized problem and works on it. The correctness of the severity assigned by the tester is verified.
- ▶ **Verifying.** The development team has delivered a fix for the problem. The tester who has opened the defect is invited to verify its resolution.
- ▶ **Rejected.** The development team doesn't recognize the defect as a real problem. Reasons for the rejection must be provided to the test team.
- ▶ **Closed.** A member of the test team, usually the tester who has opened the defect, must close it whenever it is in the rejected or verifying status. If the defect has been fixed, the tester is responsible to verify its resolution before closing it.

Even though the optimum would be to adopt an automated system, often in fast paced e-commerce projects a member of the test team is responsible for manually storing and generating reports on defect status.

6.2.3 Version control

Development teams typically need coordinated and controlled multi-user access to code and project resources: a version control mechanism applied to a common persistent repository ensures the integration and quality of interim work products.

VisualAge for Java can configure a common repository that can be synchronized with the local workspace of each team member: its version control features include resource locking and a powerful mechanism to visualize the differences between different versions of the same resource, with the possibility to choose for each difference what must be included in the new released version. For more information about version control in VisualAge for Java, refer to the product manuals.

The repository currently supported by WebSphere Studio Application Developer is CVS (Concurrent Version System). It allows resource synchronization between the developer workspace and the repository, but if conflicts arise between the resource local version and the version stored in the repository, it is necessary to perform a manual merge between the two versions. For more information about CVS, refer to the WebSphere Studio Application Developer manuals, or look at the CVS Web site <http://www.cvshome.com>.

6.3 Test phases

The testing process can be subdivided into different phases, in such a way that the output of each phase is a milestone on the test plan representing a known level of integration and quality of the developed application.

For e-commerce applications, the testing process typically goes through the following phases:

- ▶ Static test
- ▶ Unit test
- ▶ Functional test
- ▶ System test

For each of these phases, the following sections explain objectives, input (entry criteria) and outputs (exit criteria). Moreover, they identify which team members must be involved in the test execution, and describe typical types of test associated, whenever available, to useful test tools.

6.3.1 Static test

Static testing allows the test process to go through all the project phases: the results are early defects and issue discoveries which carry to less expensive and painful changes. It involves the test team in reviews for verification and validation of project plans, requirements analysis and application design: it starts taking as input drafts of the documents to be reviewed, and produces as output a completed and verified versions of them.

Also code inspections can be considered as part of the static test phase: their objective is to check for code compliance to standards and best practices adopted by the project (see the following section “Programming best practices” for some basic guidelines). Typically, inspections are carried out by developers with the most experience in the team.

Coding standards and best practices

Defining coding standards and best practices has the main purpose of leading the code developed by the team to a greater consistency: more consistent code is also easier to develop, understand and maintain, with the result of reducing the overall project time and cost.

First of all, some basic rules have to be kept in mind:

- ▶ Every person writing code for the product must follow the code conventions.
- ▶ No standard or best practice is perfect and applicable to all situations: it is important to understand when to apply standards as well as when not to apply them.
- ▶ All the deviations must be documented: when a standard or best practice is not followed, reasons and potential implications must be explained.

Note that development tools such as VisualAge for Java and WebSphere Studio Application Developer can enforce code standards compliance and make it more automatic and easy to accomplish.

The Java conventions detailed in the following sections are:

- ▶ Naming conventions
- ▶ Documentation conventions
- ▶ Programming best practices

These sections are intended to be only a summary of the most common Java conventions. More details can be found in *The Elements of Java Style*, Vermeulen, Ambler et al. Cambridge University Press, ISBN 0-521-77768-2. You can also refer to the article *Writing Robust Java Code*, by Scott W Ambler, at the Web site <http://www.ambysoft.com/javaCodingStandards.pdf>.

Naming conventions

The general guidelines to make a good name are:

1. Use meaningful and self-explanatory names (do not omit vowels!)
2. Use domain terminology, whenever available
3. Use mixed case to make names readable: use lower case in general, but capitalize the first letter of any non-initial word and for class and interface names, also of capitalize the first letter of the initial word)
4. Capitalize only the first letter in acronyms
5. Avoid long names (keep them <=15 characters)
6. Avoid names that are similar or differ only in case

Table 6-1 contains a summary of naming conventions per programming item.

Table 6-1 Naming conventions per programming item

Item	Naming Convention	Examples
Package	Use single, lower case word for package name and concatenate it to the reverse internet domain. All new package names should be agreed with the rest of the team.	com.ibm.commerce.wc54 handbook.commands
Class	Use meaningful names in mixed cases style, with the first letter capitalized. Use plurals for classes that group related attributes, services or constants.	MapMakerResourceBoun dle
Interface	Use meaningful names in mixed cases style, with the first letter capitalized. Do not use 'I' as the first letter of the Interface. Optionally, the 'able', 'ible', or 'or' postfixes can be used.	Runnable, Prompter, Singleton
Exception	Exception class names must follow the class names conventions, adding 'Exception' as the last word. Optionally, they can start with an acronym indicating the application to which they belong (e.g. EC for WebSphere Commerce).	ECSystemException

Item	Naming Convention	Examples
Method	Use a name fully descriptive of what the method does, in mixed cases style with the first letter in lower case. Try to start with an active verb whenever possible. Use the prefix 'is' for all and only the methods returning a boolean, including the boolean getters.	doProcessCategoryRelations(), isRetriable()
Argument/Parameter	Use meaningful names in mixed cases style, with the first letter in lower case. For method arguments, it can be used the prefix 'a' or 'an'.	customer, account or aCustomer, anAccount
Field	Use nouns in mixed cases style, with the first letter in lower case. Use plurals for collections. For final static fields and for fields in interfaces use all uppercase letters with the words separated by underscores. Avoid using single character for trivial local variable (such as 'c' for a char), with the exception of counters ('i', 'j', 'k').	accountNumber, COMPONENT_EXTERN

Documentation conventions

The general guidelines for a well-documented code are:

1. Comments should contain all and only information that is relevant to reading and understanding the code.
2. Comments must document what is being done and why
3. Comments must be kept as simple and short as possible
4. Avoid comments that are likely to get out of date as the code evolves

5. Avoid decoration, such as comments enclosed in large boxes drawn with asterisks or other characters
6. Write the documentation before writing the code: because you're investing time in writing documentation, and it is supposed to make your code easier to understand, try to take advantage of it during the development phase.

Table 6-2 contains a summary of usage conventions per each type of Java documentation.

Table 6-2 Usage conventions per Java documentation type

Documentation type	Usage Convention
Javadoc	Use javadoc immediately before declaration of all the interfaces, classes, methods and fields to document their purpose and description as well as their declaration.
C-style comment	Use C-style comments for all the source files general information (a project standard should be defined) and for commenting out lines of code that are temporarily not applicable.
Single line comment	Use single line comments internally within methods to document business logic, sections of code and declarations of temporary variables.

Programming best practices

Table 6-3 contains a summary of best practices in Java programming that are commonly recognized as critical to the maintainability and enhanceability of the code.

Table 6-3 Java programming best practices

Target	Best practices
Code Formatting	<ul style="list-style-type: none"> Indent nested code by 4 spaces. Do not use tabs. Break up long lines. Use blocks for all if, else, while and for statements. Put beginning brace at the end of the line that starts the block.

Target	Best practices
Classes and Interfaces	<p>Minimize the public and protected interfaces.</p> <p>Define the public interface for a class before beginning to code it.</p> <p>Define small classes with a small number of methods (<30)</p>
Fields	<p>Fields should always declared private, except for classes that are essentially data structure.</p> <p>Use always accessor member functions to access fields.</p> <p>Use final static fields or accessor member functions (preferred) for numerical constants.</p> <p>Always initialize static fields.</p> <p>Do not hide names: avoid using the same names for local variable as for class fields.</p>
Accessor member functions	<p>Whenever possible, make accessors protected and not public.</p> <p>Consider using lazy initialization for fields in the database.</p> <p>For collections, add class methods to insert and remove items.</p>
Methods	<p>Document methods both with javadoc and internal comments and specify the order of operations.</p> <p>Define small methods, normally with less than 30 statements.</p> <p>Indent the code.</p> <p>Only have one return statement per method</p> <p>Write short and single command lines.</p>
Property files	<p>All property files must have an header with a description of their content.</p> <p>Each property entry must be preceded by a comment including its purpose and the list of possible values.</p>

JTest

JTest is a tool to automatically perform static code analysis: it allows to define coding standards and metric thresholds for the size and complexity of the code, both at single class and at entire project level. It monitors coding standard compliance and metrics, signals all the violations and produces metrics reports. For more information about JTest, look at the Web site

<http://www.parasoft.com/jsp/products/home.jsp?product=Jtest>

6.3.2 Unit testing

The purpose of unit testing is to verify the internal logic and behavior of each single unit within a class. It is performed by developers throughout the whole implementation phase and is based on design documents and products reports about successes and failures occurred during the test cases execution.

Note that it is fundamental to continuously perform regression tests for the duration of the unit test phase, in order to check that the new or updated functions do not impact the already implemented ones. Test automation becomes a key factor in minimizing the required cost of time and resources.

The following sections describe some useful tools for performing complete and automated unit testing.

JUnit

JUnit is an open source testing framework for Java which is included in WebSphere Studio Application Developer and can be very useful for unit testing.

For each class to be tested with this tool, you must create a test class that inherits the JUnit framework functions from `TestCase`. Moreover, you must implement each test case to be performed by means of this class as a method containing an exit assertion to be verified. Based on the verification of such assertions, JUnit produces reports with a list of all the successes and the failures occurred during the test execution.

JUnit enables you also to create a collection of test cases by using the `TestSuite` class: for each test class added to the suite, it automatically detects and run all the test methods whose name starts with `test`: in fact, all the test methods should follow the naming convention of starting with `test` and finishing with a name fully descriptive of what the method tests. Otherwise, you can manually add to the suite methods that for some reason don't respect the naming convention. The appropriate usage of `TestSuite`, for instance as a group of

classes testing a particular package or macro-functionality, allows you to automatically run and produce reports for all the tests necessary to verify that both the new implemented functions work as designed and all the other functions have not been impacted by them.

Details about JUnit can be found at the Web site <http://www.junit.org>, whereas more information about how to install and configure JUnit in WebSphere Studio Application Developer can be found into the product manuals. Note that the provided JAR file contains also useful examples of test case classes.

JTest

JTest, a test tool already mentioned in 6.3.1, “Static test” on page 173, provides support for junit test with the possibility to run both its own test classes and the JUnit ones. You can find more information about JTest at the Web site <http://www.parasoft.com/jsp/products/home.jsp?product=Jtest>.

Unit testing for WebSphere Commerce applications

JUnit and JTest fit very well the requirements of a complete and automated unit testing in case of self-contained objects, but they are usually avoided for testing components of e-commerce applications which leverage services provided by the run-time WebSphere Commerce infrastructure (e.g. commands). In fact, in this last case it's too much an effort to implement an unit test case which can simulate such services. Moreover, it is not really significant to test the component on an environment totally different from the one where it will actually have to run.

As a consequence, unit testing of e-commerce application components is usually performed on the test environment provided by VisualAge for Java and WebSphere Studio Application Developer, where the actual WebSphere Commerce infrastructure can be imported. Besides, the integrated debugging and tracing features are highly effective tools for problem determination. More information about the WebSphere test environment can be found in Chapter 4, “Planning and development” on page 71.

In any case, JUnit and JTest can be still used for unit testing utility classes developed in order to provide services, such as complicated computations for processes output data, which you may want to keep isolated from the WebSphere Commerce infrastructure.

EJB unit testing

VisualAge for Java and WebSphere Studio Application Developer provide a client test utility that allows you to perform unit testing on the Enterprise JavaBeans by using an user-friendly graphic interface. For more information about it, refer to the *IBM WebSphere Commerce Programmer's Guide Version 5.4*.

6.3.3 Functional test

The functional test is performed by the test team with the purpose of verifying that application requirements are met by fully and separately testing each implemented function, either new or updated, in an environment independent from the development one. It is based on requirements analysis and design documents and produces test cases reports (see 6.4, “Test case design” on page 185 for details). The application test plan should put particular stress in testing new and complex customizations, such as back-end system integration, rather than small modifications.

The following is a list of test guidelines for each customizable WebSphere Commerce subsystem (for details about WebSphere Commerce subsystems, refer to 2.2.1, “WebSphere Commerce server subsystems” on page 18):

- ▶ Catalog. Test the navigation across all the categories and the proper display of the product details pages.
- ▶ Order. Test the complete order workflow. Test order cancellation and modification after submission.
- ▶ Membership. Test all the member subsystem functions such as registration forms and member approvals.
- ▶ Trading. Particular attention should be put in testing this core component of the store. Test the different trading mechanisms such as auctions, RFQ, and catalog based trading.
- ▶ Messaging. Test the correct integration with the chosen messaging system (MQ, SMTP server or FTP).
- ▶ Marketing. Test the creation of marketing campaigns, electronic coupons, discounts, collaboration, and so on.

Regression and test automation

Time constraints usually limit the functional regression testing to a significant subset of test cases. Again, test automation is fundamental. The following is a list of test tools for automated response analysis in order to verify whether the test has passed (for more details, refer to the “Testing guidelines” chapter in *WebSphere Commerce V5.4 Handbook Architecture and Integration Guide*, SG24-6567):

- ▶ Segue Silktest allows to test your Web site using a wide sets of Web clients combined with several operating system. More details can be found at the Web site <http://www.segue.com>
- ▶ Mercury Interactive WinRunner can be used by any Windows application: it allows to specify expected responses in several checkpoints, and to compare

compare them with the actual ones. More details can be found at the Web site <http://www.mercuryinteractive.com/products/winrunner>.

6.3.4 System test

System testing verifies that all components, cooperate properly to meet the business requirements. It is performed by the test team in an environment as similar as possible to the production one. It is based on requirements analysis and design documents and produces test cases reports (see 6.4, “Test case design” on page 185 for details).

Different types of test can be performed during the system test phase, according to the focus areas identified in the testing strategy. The following sections describe the most common types of test for e-commerce application.

System integration

As noted above, system integration test is very important for e-commerce applications which are usually the result of the cooperation and integration between several applications, residing on different systems and implemented with different technologies. Even though testing back-end systems integration starts during the functional test phase, usually only during the system test it is possible to perform integration test in a production-like environment, including network and systems configuration.

No automation tools are available for this type of test.

Scalability and performance

Scalability and performance are such a critical issue for e-commerce applications that adequate measurements should start from the functional test phase, in order to verify that the application design, architecture and implementation meet the requirements. However, it is only during the system test phase that such measurements become really indicative of what will happen in the actual production environment.

Testing scalability and performance usually consists of stimulating high loads on the server and of measuring both the end-user experienced response time and the server resources usage.

The following is a list of available tools for performing each of the above test tasks on WebSphere Commerce applications (for more details, refer to the “Testing guidelines” chapter in *WebSphere Commerce V5.4 Handbook Architecture and Integration Guide*, SG24-6567):

- ▶ Generating Web sites load

- Segue SilkPreview is shipped with WebSphere Commerce V5.4 and allows to specify up to five different URLs to be requested accordingly to a configured frequency.
- OpenSTA (Open System Testing Architecture) executes test cases scripts making them run concurrently by a configured number of virtual users.
- ▶ Measuring WebSphere Commerce performance
 - WebSphere Commerce Performance Monitor periodically monitors items as commands, URLs and Views (JSP) for a WebSphere Commerce server instance.
 - WebSphere Resources Analyzer monitors both application resource (such as EJBs, servlet session manager and Web application instance) and WebSphere resources (such as Java Virtual Machine, database connection pool, transaction manager and application server thread pools).
 - DB2 has internal monitoring tools, such as Snapshot Monitor, Event Monitor, Explain facility, db2batch tool, CLI/ODBC/JDBC Trace Facility (for details, refer to the redbook *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012).
 - JVM Profiler tools monitor garbage collection and memory leaks, performance bottleneck and deadlocks, threads interaction and objects relations. An example of such tools is Jinsight (for details, refer to the Web site <http://www.alphaworks.ibm.com/tech/jinsight>).
- ▶ Testing the end-user experience of the online store
 - Page Detailer, shipped with WebSphere Application Server, Advanced Edition, monitors the overall response time for each page of the store, and retrieves information about timing, size and identity for each item inside the page (for details about Page Detailer, refer to the Web site <http://ibm.com/software/webservers/studio/doc/v35/pagedetailer/EN/HTML>
For more information about Web design guidelines, refer to the article Design for performance, that can be found at the WebSphere developer domain Web site:
<http://www7b.software.ibm.com/wsd/library/techarticles/hvws/perform.html>

Security

The purpose of testing e-commerce application security is to assure that confidential information is protected against loss, corruption and misuse either by deliberate or accidental actions.

The following is a list of tools to test security in WebSphere Commerce applications, divided by security areas (for more details, refer to the “Testing guidelines” chapter in the redbook *WebSphere Commerce V5.4 Handbook Architecture and Integration Guide*, SG24-6567):

- ▶ In order to keep the operating system updated with the last security fix packs, the WebTrends NetIQ Security Analyzer tool can be used for the Windows, Sun Solaris and Red Hat Linux. More information can be found at the Web site <http://www.Webtrends.com/products/wsa>
- ▶ In order to scan Web servers log files for Intrusion Detection and simulate attacks that can cause a store Denial of Service, you can use as test tools:
 - Tivoli Intrusion Manager (refer to the Web site http://www.tivoli.com/products/index/intrusion_mgr/)
 - Computer Associates eTrust Intrusion Detection (refer to the Web site <http://www3.ca.com/Solutions/Solution.asp?ID=271>)
 - Mercury Interactive ActiveTest SecureCheck (refer to the Web site <http://www-svca.mercuryinteractive.com/products/securecheck/>)
- ▶ In order to check and delete WebSphere Commerce temporary files that may contain potential security exposures, the WebSphere Commerce Security Checker tool can be configured as a scheduled task, that by default run once a month (for more details, refer to the “Security design guidelines” chapter in the redbook *WebSphere Commerce V5.4 Handbook Architecture and Integration Guide*, SG24-6567).

Usability

Usability is a key factor for a successful online store: easy to navigate stores can capture the attention of a wider range of potential Web customers, including those Web navigators completely new to e-commerce and online stores.

Usability tests must be designed by testers but they must be performed by potential end users with different levels of expertise in using Web sites: testers must prepare a list of the typical tasks that store customers can perform, and users must try to perform such tasks reporting the required amount of time as well as the errors made and the difficulties found (a deep analysis of Web usability can be found in *E-Commerce User Experience*, Jakob Nielsen, Rolf Molich et al. Nielsen Norman Group, ISBN: 0-9706072-0-2).

Regression

System test must be concluded with a final regression test performed when all the functions are stabilized and the code is frozen. Time constraints usually limit this last regression to a significant subset of test cases: it is very important to use test tools, such as those listed above, in order to automate this test and to optimize the cost of time and resources.

6.4 Test case design

Test case design must be focused on testing strategy, types of test and test tools. Moreover, it must be based on requirements analysis and design documents. As part of test planning, a test case document template should be produced, containing the following sections:

- ▶ Objectives. This is a brief summary of the test case objectives, according to the general testing strategy and test plan.
- ▶ Environment. This a description of the hardware, software and network configuration adopted for the test execution.
- ▶ Input. This is the list of all the inputs necessary to run the test. For each input, it must be specified also the actual value to be used, in order to enable testers to repeat the test and compare the obtained results.
- ▶ Description. This is a step-by-step description of all the tasks to be performed during the test execution.
- ▶ Test tools. This is a list of the test tools used to execute the test.
- ▶ Expected results. This is the definition, as precise as possible, of the results expected by the test case execution.
- ▶ Actual results. This is the actual test case report, containing a description of the obtained results, with references to defects and issues discovered during the test case execution, each one associated to its current status (see 6.2.2, “Problem management” on page 171 for details about defects statuses). A test case can be considered successfully executed and completed if all its defects have been closed.

Note that it is very important to design test cases both for normal and for error conditions, in order to verify that they are handled by the system and that they bring to expected results.

The next section contains some samples of test cases.

6.4.1 Test cases samples

This section contains some samples of system test cases, provided in form of tables, which have been designed in order to test customizations implemented for the store requirements described in this redbook.

Test CICS order transactions

Table 6-4 and Table 6-5 contain system test cases designed for the customized code implementing CICS order transactions, respectively in normal and in error conditions. For more information about the CICS order transaction implementation, and for a complete explanation of the environment set-up and the deploy of the customizations, refer to Chapter 9, “Orders” on page 233.

Table 6-4 Test case for CICS order transactions in normal conditions.

Test item	Description
Objectives	Test the order CICS transaction function in normal conditions.
Environment	<p>The environment set-up include:</p> <ul style="list-style-type: none"> ▶ WebSphere Commerce 5.4 server running on W2000 and containing: <ul style="list-style-type: none"> – an instance of the ToolTech store – the ECI J2EE related classes – a record in the STADDRESS table that looks like: <ul style="list-style-type: none"> • address1: tcp://gunner.almaden.ibm.com • memberid: -2002 • address2: SCSCPAA6 • address3: ECIPROGX • business title: ToolTech • nickname: CTG-PROP. ▶ CICS server running on zOS. ▶ CICS Transaction Gateway V4.0.1 running on Windows 2000 Server (in the example the gateway ip address is: gunner.almaden.ibm.com). The CICS Transaction Gateway set up enables access to the CICS region (server) SCSCPAA6.
Input	A product order. Note that the information sent to the CICS server are the id and total price of the order.
Description	<ol style="list-style-type: none"> 1. Enter the ToolTech store. 2. Go to the product details page. 3. Enter quantity=1 and add to the shopping cart. 4. Check out the order. 5. Click Order Now in the order summary page. 6. Look in the order confirmation page for the order number.

Test item	Description
Test tools	An Internet Explorer 5.5 browser
Expected results	<p>The CICS server sends back for acknowledgement the following transaction information, which must be stored both in the ORCOMMENT table and in the WebSphere Commerce instance log file.</p> <ul style="list-style-type: none"> ▶ Application Id = SCSCPAA6 ▶ Date ▶ Time in hh:mm:ss format (<i>check the presence of both the colons!</i>)
Actual results	<p>The test passed all the conditions:</p> <ul style="list-style-type: none"> ▶ the ORCOMMENT table contains the following record: <ul style="list-style-type: none"> – ORCOMMENT_ID: 10002 – ORDERS_ID: 10257 – LASTUPDATE: 2002-05-10 11:40:59.062 – COMMENTS: SCSCPAA6 10/05/02 11:38:50 ▶ the WebSphere Commerce instance log file contains the following entry: <pre> ===== TimeStamp: 2002-05-10 11:40:58.765 Thread ID: <Server Thread> Class: DoCicsTransJ2EECmdImpl Method: performExecute Severity: 16 Message Text: CMN1519I Generic informational message: "CICS Transaction information for orderId=10257: CICS Trans response='SCSCPAA6 10/05/02 11:38:50 '". </pre>

Table 6-5 Test case for CICS order transactions in error conditions.

Test item	Description
Objectives	Test the order CICS transaction function in error conditions: simulate the CICS server down situation by setting a wrong IP address into the STADDRESS table.
Environment	<p>The environment set-up is the same as in the previous case (see Table 6-4). Only the value into the address1 column of the STOREADDRESS table must be changed as follows:</p> <p>address1: <i>tcp://gunner.almaden.ibm.com1.</i></p>

Test item	Description
Input	A product order. Note that the information sent to the CICS server are the id and total price of the order.
Description	<ol style="list-style-type: none"> 1. Enter the ToolTech store. 2. Go to the product details page. 3. Enter quantity=1 and add to the shopping cart. 4. Check out the order. 5. Click Order Now in the order summary page. 6. Look in the order confirmation page for the order number.
Test tools	An Internet Explorer 5.5 browser
Expected results	<p>The WebSphere Commerce cannot connect to the CICS server, with the following results:</p> <ul style="list-style-type: none"> ▶ the browser shows the generic error page explaining that the store is experiencing some problems (there is no need to make customers aware that the CICS server is down). ▶ the ORCOMMENT table does not contain any new record because no transaction has occurred. ▶ the WebSphere Commerce instance log file contains an entry explaining the occurred problem.

Test item	Description
Actual results	<p>The test passed the following conditions:</p> <ul style="list-style-type: none"> ▶ the browser showed the generic error page. ▶ the ORCOMMENT table does not contain any new record. ▶ the instance log file contains the following entry: <pre> ===== TimeStamp: 2002-05-10 11:53:29.218 Thread ID: <Servlet.Engine.Transports:10> Class: DoCicsTransJ2EECmdImpl Method: doTransaction Severity: 1 Message Text: CMN0409E The following error occurred during processing: A Resource exception occurred when trying to connect to:{URL=tcp://gunner.almaden.ibm.com1, PROG=ECIPROGX, REGION=SCSCPAA6}. Exception: javax.resource.spi.CommException: CTG9630E: IOException occurred in communication with CICS. [..... the Exception stack trace]. </pre>

Test loading products via MQSeries

Table 6-6 and Table 6-7 contain system test cases designed for loading products via MQSeries, respectively in normal and in error conditions. For more information about the implementation, and for a complete explanation of the environment set-up and the deploy of the customized code, refer to Chapter 11, “Messaging customization” on page 291.

Table 6-6 Test case for loading products via MQSeries in normal conditions.

Test item	Description
Objectives	Test loading products via MQSeries in normal conditions.
Environment	<p>The environment set-up include:</p> <ul style="list-style-type: none"> ▶ WebSphere Commerce 5.4 server running on W2000 and containing: <ul style="list-style-type: none"> – an instance of the WebFashion store – the MA88 Product Extension Pack ▶ MQSeries 5.2.1 server running on the same W2000 machine as WebSphere Commerce.

Test item	Description
Input	An XML message describing a product identified as "MyProduct" to be created in the "Outerwear" subcategory of the "Men's" category (for details, refer to 11.5.1, "Defining XML and DTD for the command" on page 300).
Description	<ol style="list-style-type: none"> 1. Put the XML content on the "wcs_inbound_ser" queue via the MQSeries explorer. 2. Enter the WebFashion store. 3. Browse the "Outerwear" subcategory in the "Men's" category. 4. Verify that the MyProduct is properly shown on the page.
Test tools	<ol style="list-style-type: none"> 1. An Internet Explorer 5.5 browser. 2. The MQSeries 5.2.1 client.
Expected results	The product and its items are properly shown on the web page with their attributes.
Actual results	The test passed all the conditions.

Table 6-7 Test case for loading products via MQSeries in error conditions.

Test item	Description
Objectives	Test loading products via MQSeries in the error condition of an inbound XML message describing a product to be created into a non-existing category.
Environment	The environment set-up is the same as in the previous case (see Table 6-6).
Input	<p>The same XML message as in the previous case (see Table 6-6), but with the category identifier changed from "Outerwear" to "Outerwaer", in the following way:</p> <pre><categoryrelations> <categoryrelation categoryidentifier="Outerwaer" catalogidentifier="WebFashion" sequence="0" action="create"/> </categoryrelations></pre>

Test item	Description
Description	<ol style="list-style-type: none"> 1. Put the XML content on the "wcs_inbound_ser" queue via the MQSeries explorer. 2. Enter the WebFashion store. 3. Browse the "Outerwear" subcategory in the "Men's" category. 4. Verify that the MyProduct is not shown on the page.
Test tools	<ol style="list-style-type: none"> 1. An Internet Explorer 5.5 browser. 2. The MQSeries 5.2.1 client.
Expected results	<ul style="list-style-type: none"> ▶ The product is not loaded. ▶ the WebSphere Commerce instance log file contains an entry explaining the occurred problem.
Actual results	<p>The test passed the following conditions:</p> <ul style="list-style-type: none"> ▶ The product is not shown in the browser ▶ the instance log file contains the following entry: <pre> ===== TimeStamp: 2002-05-20 17:29:57.781 Thread ID: <Thread-10> Class: com.ibm.commerce.wc54handbook.com- mands.MQProductCategoryRelationsCmdImpl Method: getCatgroupId Severity: 1 Message Text: CMN0409E The following error occurred during processing: Error when retrieving catgroupId by identifier Outerwaer and memnberId 70000000000000000001. </pre>

6.5 Test environment set-up

In order for the testing to be effective, we recommend that you set up a test environment independent from both the development and the production environments, but as similar as possible to production.

Moreover, after your store have been published and is working online, when the maintenance phase is entered, then the test environment allows the site administrator to perform verifications and measurements without affecting the production system, and to test any necessary change of product catalogs or shopping functions before publishing it into the online store.

WebSphere Commerce provides support for setting up a test environment in which a staging server works with a test data base: the following section describes these components and explains how they are related to the correspondending production components.

6.5.1 Staging server and test data

A WebSphere Commerce staging server is a WebSphere Commerce instance that has the same configuration of the production instance, but runs on a separate machine and is connected to a test data base.

Any WebSphere Commerce instance can be set up as a staging server, both during and after installation: for more information, refer to the *IBM WebSphere Commerce for Windows NT and Windows 2000 Installation Guide for use with DB2 Universal Database Version 5.4* and to the WebSphere Commerce Version 5.4 Online Help.

The test data base contains the same schema as the production database, but includes an additional set of tables for publishing purposes and a set of triggers to log all the changes performed on the database.

The WebSphere Commerce staging utilities allow site administrators to copy their production databases to test databases and to propagate changes tested on the staging environment into the production system, according to the information flow shown in Figure 6-1 on page 193.

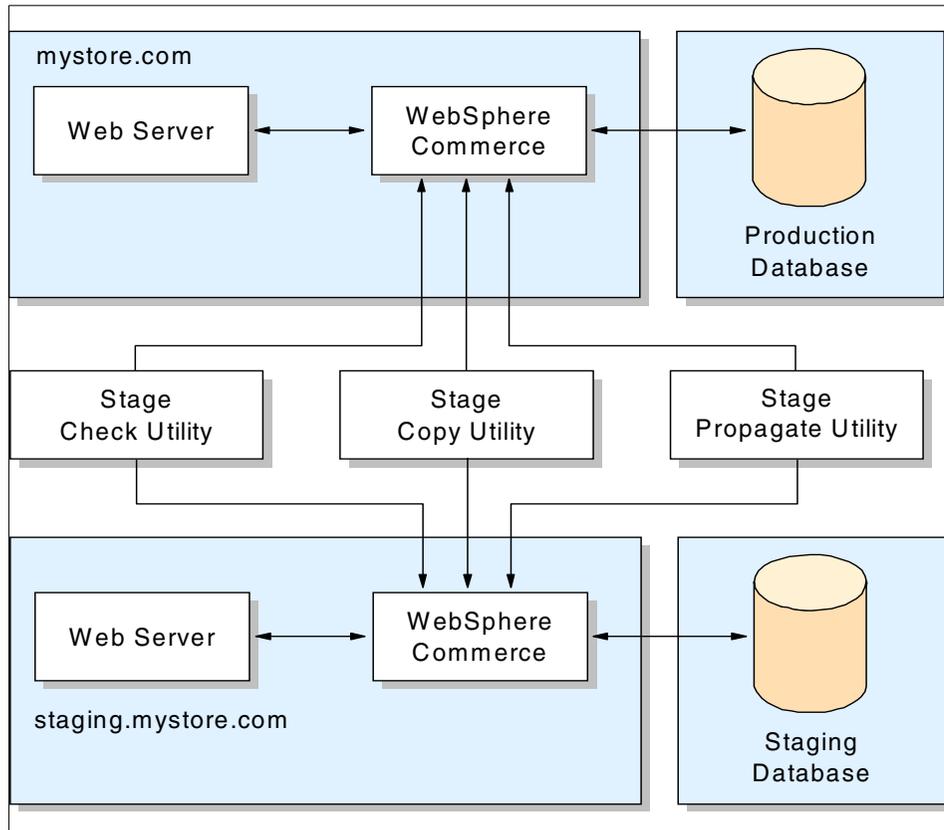


Figure 6-1 Staging server configuration and utilities.

The following sections contain a brief description of each of the WebSphere Commerce stage utilities shown in the above diagram. For more information, refer to the *WebSphere Commerce Version 5.4 Online Help*.

Stage Check utility

The Stage Check utility allows the site administrator to identify and correct any potential conflict of unique indexes before propagating your changes to the production database.

Moreover, you can customize and extend this utility by adding more tables, even your customized tables, to its check list.

Stage Copy utility

The Stage Copy utility copies data from the production to the staging database allowing site administrators to specify as targets individual as well as site-related and merchant-related tables.

Stage Propagate utility

The Stage Propagate utility performs the actual propagation of changed data from the staging to the production database.

The STAGLOG table in the test database contains information about records to be inserted, updated or removed in the production database: already propagated changes are flagged as processed with a '1' in the STGPROCESSED column.

Note that the Stage Propagate utility can only propagate changed data in the production database: schema changes such as new indexes or tables must be manually executed on it by means of the command utilities contained into the WebSphere Commerce Loader package.

Moreover, any Web resources referenced by staging records, such as images, HTML or JavaServer Pages files, must be manually copied from the staging to the production system.

6.6 Problem determination

Problem determination is the very first step to solve problems which can occur both in the development and in the maintenance phases of the store.

The WebSphere Commerce logging service supports site administrators by storing information about the commerce server activity and by notifying them whenever errors or abnormal conditions occur. It logs two types of information:

- ▶ Diagnostic data, which are stored in a log file for problem determination purposes. Tracing is an example of diagnostic log.
- ▶ Activity data, which are stored in the WebSphere Commerce database to record system events.

The logging service's components are:

- ▶ ECTrace. It traces the run-time execution path of the store components. It is used for problem determination purposes by the store development and technical support teams.
- ▶ ECMessageLog. It writes into the log file diagnostic messages which help in problem determination the development and technical support teams as well as the site administrator

- ▶ `UserTrafficEventListener`. It stores into the WebSphere Commerce database data about activities performed on the system to be used by the site or store administrator.

Site administrators can configure logging by using the Log System panel of the Configuration Manager or directly by editing the commerce instance XML configuration file. For logging configurations that apply only to the instance current life cycle, they can also use the WebSphere administrative console.

The logging service configurable parameters are:

- ▶ Trace and log file paths (they can have the same value if you want to merge trace and log data into the same file)
- ▶ Trace and log maximum sizes (a new file is created every time one of these maximum sizes is exceeded, so remember to set both to the same value, if you want to have trace and log data always stored in the same file)
- ▶ Tracing and logging enablement (only error messages are logged by default).
- ▶ Filters by component to be applied to trace data
- ▶ Filters by severity to be applied to log data (possible values: error, warning, status, debug, informational)
- ▶ Maximum size of the activity log's cache
- ▶ Errors notification mechanism enablement and configuration.

Note that only the strictly necessary amount of data should be logged, in order not to impact uselessly the system performance. For more information about logging service components and configurations, refer to the *WebSphere Commerce Version 5.4 Online Help*. Moreover, in order to perform a more effective problem determination on your store, you must understand how to use the error handling and tracing mechanism in your customized code. The following sections provide some guidelines and samples. For more details see the *IBM WebSphere Commerce Programmer's Guide Version 5.4*.

6.6.1 Tracing

Tracing allows people of the development and technical support teams to debug the store, that is to log the application data flow in order to understand the executed path and determine which problems caused the experienced malfunction.

As already noted, tracing can be filtered by component and must be enabled only whenever it is necessary.

The WebSphere Commerce tracing service is performed by the ECTrace class in the com.ibm.commerce.ras package.

In order to use tracing in your customized code, you have to:

1. Enable for tracing the EXTERN component, using the Configuration Manager
2. Set the trace entry point
3. Set trace point whenever necessary, specifying the text to be traced
4. Set the trace exit point

Example 6-1 shows as ECTrace has been used within the performExecute() method of the MQProductCreateCmdImpl class, implemented for the customization described in Chapter 11, "Messaging customization" on page 291 .

Example 6-1 performExecute() method of the MQProductCreateCmdImpl class

```
public void performExecute() throws ECEException {
    String methodName = "performExecute";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
    super.performExecute();

    // call doProcess() for every product in the productVector
    for (int j = 0; j < getProductVector().size(); j++) {
        TypedProperty aProduct = (TypedProperty)getProductVector().elementAt(j);
        setRequestPropertiesPerProduct(aProduct);
        if (!doProcess(super.requestProperties))
            throw new ECApplcationException(ECMessage._ERR_DO_PROCESS,
                getClass().getName(), "performExecute");
        cleanUp();
    }
    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}
```

Example 6-1 shows the trace information captured into the WebSphere Commerce instance log file during the preceding performExecute() method invocation.

Example 6-2 Trace information related to the performExecute() method execution.

```
=====
TimeStamp:    2002-05-06 12:27:48.796
Thread ID:    <Thread-7>
Component:    EXTERN
Class:        com.ibm.commerce.wc54handbook.commands.MQProductCreateCmdImpl
Method:       performExecute
Trace:        ENTRY POINT
=====
```

```
TimeStamp: 2002-05-06 12:27:48.796
Thread ID: <Thread-7>
Component: EXTERN
Class: com.ibm.commerce.wc54handbook.commands.MQProductCreateCmdImpl
Method: setRequestPropertiesPerProduct
Trace: ENTRY POINT
```

```
=====
TimeStamp: 2002-05-06 12:27:48.796
Thread ID: <Thread-7>
Component: EXTERN
Class: com.ibm.commerce.wc54handbook.commands.MQProductCreateCmdImpl
Method: setRequestPropertiesPerProduct
Trace: EXIT POINT
```

```
=====
TimeStamp: 2002-05-06 12:27:56.703
Thread ID: <Scheduler default#1>
Component: ORDER
Class: com.ibm.commerce.payment.commands.PaySynchronizePMCImpl
Method: performExecute
Trace: ENTRY POINT
```

```
=====
===== Other trace information from other methods.=====
=====
```

```
TimeStamp: 2002-05-06 12:34:06.468
Thread ID: <Thread-7>
Component: EXTERN
Class: com.ibm.commerce.wc54handbook.commands.MQProductCreateCmdImpl
Method: doProcessAttributes
Trace: EXIT POINT
```

```
=====
TimeStamp: 2002-05-06 12:34:06.468
Thread ID: <Thread-7>
Component: EXTERN
Class: com.ibm.commerce.wc54handbook.commands.MQProductCreateCmdImpl
Method: performExecute
Trace: EXIT POINT
```

6.6.2 Error handling

In WebSphere Commerce error handling is tightly integrated with the logging service so that whenever an exception is thrown, it is automatically logged.

Error handling can be performed by both JSPs and commands:

- ▶ JSPs can directly handle database exceptions and take appropriate recovery actions, or try rollbacks of current failed transactions using their own, or application default, JSP error templates.
- ▶ For commands, WebSphere Commerce provides an easy-to-use and multicultural stores supporting error handling framework that will be described in the following.

Command exceptions are caught by the Web Controller: such exceptions can be of two different types:

- ▶ `ECApplicationException`. It is thrown for user errors, typically invalid input parameters. The command won't be retried by the Web Controller.
- ▶ `ECSystemException`. It is thrown for configuration errors or runtime exceptions. The Web Controller will retry a retrievable command in case of database deadlocks or rollbacks.

A command exception must contain the following information:

- ▶ Error view name. The Web Controller looks for it in the VIEWREG table and invoke the associated error view command.
- ▶ Error message. It is an `ECMessage` object containing info such as exception severity and type, as well as key and resource bundles to find the message text (more details in section, "Messages" on page 200).
- ▶ Error parameters. Name-value pairs for parameters to be inserted into the error message.
- ▶ Error data. Data that the JSP error template can get from the error data bean.

The following sections contain an explanation of the exception handling run-time flow, as well an overview of the WebSphere Commerce messages and a very high-level description of the basic steps to follow in order to develop a customized exception handling logic. For more details, refer to *IBM WebSphere Commerce Programmer's Guide Version 5.4*.

Exception handling flow

This section contain a summary of the exception handling flow, represented in Figure 6-2 on page 199.

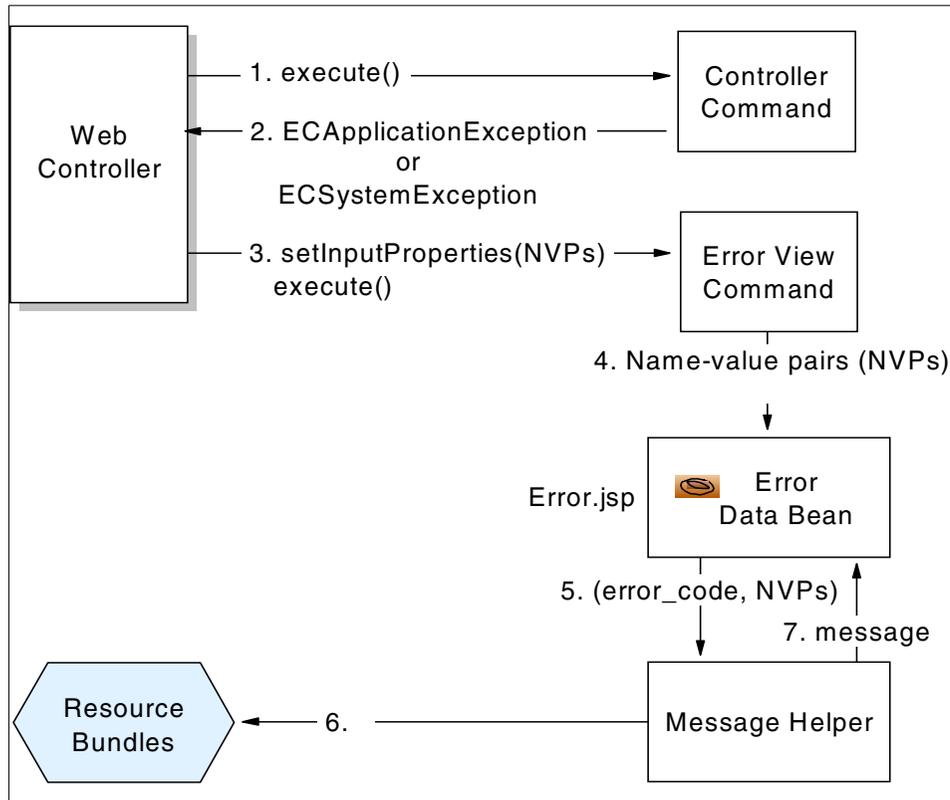


Figure 6-2 The WebSphere Commerce exception handling flow.

1. The Web Controller invokes a controller command
2. The command throws an ECApplcationException or an ECSystemException that is caught by its invoker.
3. The Web Controller gets from the exception the Error view name, in order to find in the VIEWREG table which error view command must be invoked. Moreover, it passes to the error view command as input properties (NVPs) the other exception attributes, such as the ECMessage object, error parameters and optional error data.
4. The view command invokes the associated JSP error template passing to it the input properties (NVPs). The JSP template activates the databean populating it with the input properties (NVPs).
5. The error databean invokes the message helper object passing to it the input properties (NVPs).

6. The message helper uses the `ECMessage` object attributes to find out the message text in the appropriate resource bundles and composes the message using both the message text and the error parameters.
7. The message helper returns the message to the JSP template.

Messages

WebSphere Commerce stores the text of error messages in resource bundles, usually implemented as properties file, in order to make their maintenance easier and to support multilingual stores. For instance, the WebSphere Commerce server messages are stored in the `ecServerMessages_XX_XX` file, where `_XX_XX` indicates the locale (e.g. `_en_us`). The client locale is stored in the command context: based on it, the Web Controller determines in which properties file must be searched the required message.

WebSphere Commerce uses two types of messages:

- ▶ User messages are displayed in the browser as a result of invalid user input or invalid application state. New user messages can be created using a separate resource bundles.
- ▶ System messages are automatically stored in the message log as a result of a system malfunctionality. They are predefined and cannot be customized.

In order to create a new message, you must:

1. Create a new resource bundles, usually implemented as a property file in the `/stores/lib` directory, containing every new defined key-message text pair.
2. Create new class containing the `ECMessage` object declarations, one for different message key. Besides the key, each object must have assigned a severity (error | warning | status | debug | informational), the message type (user | system) and the resource bundles.
3. Create a new class for the message keys declarations.

Exception handling in customized code

the following steps must be followed in order to develop a customized exception handling logic:

1. Catch the Exception
2. Create the appropriate type of Exception, either `ECApplcationException` or `ECSytemException`
3. Create a new customized message.



Packaging and deployment

The *IBM WebSphere Commerce Store Developer's Guide Version 5.4* goes into great detail about most parts of the store assets needed to build a store archive. However, there are some aspects of store creation and maintenance that need to be examined even further.

7.1 Overview

The task of packaging and deploying a store is not just something that is done when the development phase is over. During the development phase it is often necessary to perform incremental builds of the store. Having an integration server with a recent build of all the different parts of a new store, helps developers and testers to test and monitor the progress of the development.

Since the WebSphere Test Environment on a development workstation runs very slowly compared to a real WebSphere Application Server, testers will need a real WebSphere Application Server on which to perform their test scenarios.

With that in mind it is important to establish a simple and repeatable packaging and deployment procedure. We will be looking at different approaches to packaging and deployment.

7.2 Using WebSphere Studio and Store Services

WebSphere Commerce comes packaged with WebSphere Studio. WebSphere Studio can import a store archive and enable the developer to modify JSPs and images and then repackage the store archive. But that is as far as WebSphere Studio goes. It will not let you modify the store archive database assets or the property files. This means that if you use WebSphere Studio to create your JSP files, then you would still have to extract the contents of the SAR file to be able to modify the property files and database assets.

Store Services has a new feature in Version 5.4 that allows you to download the webapp.zip file from the published store. You can make your modifications and upload the webapp.zip file again with the changes that you've made. Although this is a nice feature, it will not allow you to modify the property files for republishing.

Store Services will let you modify some of the database assets through the Web interface but not all.

Thus WebSphere Studio and the editing features of Store Services are tools that restricts the user to only a part of the store archive.

7.3 Using Ant to package and deploy a store

Due to the fact that not all assets created for a new store can be stored in the SAR file and since the existing tools will not give us access to all parts of the SAR file, it is necessary to set up a robust and repeatable build process that will ensure that all the assets needed by a store will be deployed in one operation.

This is especially the case during the store development process. To be able to code and test the store fast and efficiently, the development team needs to be able to deploy and redeploy updated code often. This calls for an automated build process.

When we are using the word *build*, we do not mean the actual compilation of the java source code, but instead the building of the SAR file and the deployment of the different kinds of site assets.

We have found that the build tool *Ant* will do the job of deploying and redeploying all the assets needed by a store. Since Ant uses the General Public License, you can download it free of charge on the following URL:

<http://jakarta.apache.org/ant/index.html>

For installation and configuration of Ant you should follow the online installation guide on:

<http://jakarta.apache.org/ant/manual/index.html>

For the examples used in this chapter we used the Ant windows binary Version 1.4.1.

7.3.1 Folder structure

Whether you are developing your store based on an existing store archive or creating your own from scratch, you will want to extract the contents of the store archive to enable you to modify all of the assets including database assets, JSPs, images and property files.

You must also create folders that contain the any customized code exported from VisualAge for Java in JAR files.

Additionally if you have customized any site level data you should create a folder containing the XML files for this purpose.

If you have created new database tables you should create a folder that contains the SQL files with the DDL for the tables.

In this chapter we create a src folder containing all the assets mentioned above. In our case the src directory resides in the toplevel folder called project.

We have created a site and SAR folder in the folder hierarchy just below the src folder. The SAR folder contains all the assets that constitute the store archive. The site folder contains all the site level assets.

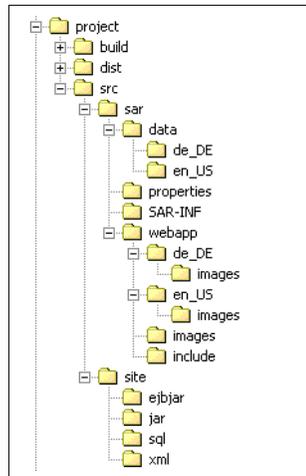


Figure 7-1 The folder structure

7.3.2 Using the sample Store Archives

If you have used one of the sample archives as a template for your store there are some modifications you need to make.

In WebSphere Commerce version 5.4 when using the Store Services to create to store you supply it with some information that it uses to customize the store archive. They are:

- ▶ Store Directory

Store Services take whatever you write here and updates the store.xml file in the store archive with that value. The value is inserted in the directory attribute of the <store> tag and in the identifier attribute of the <storeent> tag

When you are not using Store Services to do the initial deployment of your store, you will need to update these fields yourself.

- ▶ Store Owner

Store Services updates the contract.xml file with information on the store owner when its doing the initial publishing of the store archive.

When you create your own store and are not using the Web-based Store Services tool, you will need to update this file manually.

You should replace all occurrences of the entity &MEMBER_IDENTIFIER; with the name of the owner organization. You should replace all occurrences of the entity &STORE_IDENTIFIER; with the name of the store. This must be the same name as the identifier attribute mentioned above.

When store services modifies the SAR files it also copies the files DBLoadMacros.dtd into the SAR file. You need to copy the DBLoadMacros.dtd from drive:\WebSphere\CommerceServer\xml\sar into your data folder.

Modify the DBLoadMacros.dtd so that the &MEMBER_ID; entity is equal to the id of the owner organization of your store for example.: 7000000000000000001 .

If you used a sample store archive it is advisable to modify the default names of the different database entries. As an example if you used InFashion as a template you should change the trading position name from InFashion to a name you specify. Look through all the different database assets and make the appropriate changes. Doing this will also familiarize you with what kinds of data is contained in the different files.

7.3.3 The Deployment Use Case

The following use case describes what exactly it is that we wish to accomplish with the packaging and deployment.

Prerequisites

WebSphere Commerce is installed and you have created an instance. The WebSphere Application Server and HTTP server are started, and the WebSphere Commerce Enterprise application is running.

The assets to be deployed are in place in a source folder as described above. They are:

- ▶ Site Assets
 - Commands, data beans and Enterprise JavaBeans packet in JAR files
 - XML files containing for example new languages
 - SQL files to create new database tables
- ▶ Store Assets
 - JSP files and images
 - Property files
 - The SAR-INF folder
 - The DATA folder containing XML files for the store

Ant is installed and a build.xml file is customized to our needs.

Description

Executing Ant with the correct build.xml files will perform the following.

1. Execute the SQL files.

Use the DB2 command line tool to execute the SQL commands.

2. Execute the site level XML files.

Use the massload.cmd command to import the XML into the database.

3. Commands and data beans

Unjar and rejar the JAR files according to the *IBM WebSphere Commerce Programmer's Guide Version 5.4*

Copy the JAR files containing commands and data beans to:

`drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_instanceName.ear\wcstores.war\WEB-INF\lib`

4. Enterprise JavaBeans

Since the process of deploying Enterprise JavaBeans includes the use of the GUI tool Application Assembly Tool the deployment of Enterprise JavaBeans cannot be completely automated with Ant.

Ant will take care of executing the edbdeploy tool to generate WebSphere Application Server 4.0 deployed code for the Enterprise JavaBeans.

Ant will also take care of executing the modifyIsolationLevel command for the Enterprise JavaBeans.

The user will have to manually export the current Enterprise Application and export configuration information for the Enterprise Application.

The user will then use the Application Assembly Tool to add new entity beans to the Enterprise Application archive.

And to finish off the user will manually remove the existing Enterprise Application from WebSphere Application Server and import the new and modified Enterprise Application with XMLConfig.

5. SAR file

- a. Zip the JSP files and images into the webapp.zip file.

- b. Zip the properties into the properties.zip file.

- c. Zip the data and SAR-INF folders with the webapp.zip and properties.zip files into the storename.sar file.

- d. Copy the SAR file to:

`drive:\WebSphere\COMmerceServer\instances\instanceName\sar`

- e. Execute `publishstore.bat` from the command line to publish the store.

Outcome

Assuming the entire process went along without errors, the outcome of this use case will be that:

- ▶ New tables for customization have been created.
- ▶ Custom site level data has been imported into the database.
- ▶ Customized commands, databeans and Enterprise JavaBeans that are needed for the store to work properly are deployed.
- ▶ The store assets have been packaged into a SAR file and published.

7.3.4 Creating the Ant build files

Ant needs to know what to do with all your site and store assets. Ant uses a build file for this. We use that file to instruct Ant down to the smallest detail what it's supposed to do with each asset. Each build file has one project and in that project are one or more targets that contain the tasks to be executed.

All the different properties that are needed to build the project are defined in a separate file that we name `project.properties`. The property file is loaded in the beginning of the build file. To refer to a property the following format is used: `${propertyName}`. To make the deployment as flexible as possible we will create a separate build file for the different targets

Example 7-1 build.xml

```
<?xml version="1.0" ?>
<project default="init">
  <property file="project.properties" />
  <target name="init">
    <antcall ...
  </target>
  <target name="A">
    <sometask />
    ...
  </target>
</project>
```

Load custom tables

The first project that we will be making is, according to the use case, one that will take care of executing the SQL files needed to create custom tables in the database.

We have written a small batch file called `executeSQLFiles.bat` that can be executed manually from the command line. See Example 7-2. Since we want the whole process to be executed automatically through Ant, we create a target in Ant that contains a task that will execute the batch file with the appropriate command line parameters. In this example the batch file is called `loadtables.bat`. See Example 7-3. The command line parameters that we need are the name of the database, database user name and database password. These parameters are defined as properties.

Example 7-2 executeSQLFiles

```
<project default="executeSQLFiles">
<property file="project.properties" />
<target name="executeSQLFiles">
  <exec executable="db2cmd.exe" dir="${ProjectDir}/src/site/sql">
    <arg value="db2c1p" />
    <arg value="loadtables.bat" />
    <arg value="${DataBase}" />
    <arg value="${DBUserName}" />
    <arg value="${DBPassword}" />
  </exec>
</target>
</project>
```

Example 7-3 loadtables.bat

```
@echo off

db2 connect to %1 user %2 using %3
db2 -tvf tables.sql
db2 terminate

exit
```

Massload site XML

We need to write a target that takes care of massloading any site level data that you have created. This target is similar to the previous target since it is also merely executing a command.

Example 7-4 massloadSiteXML

```
<project default="massloadSiteXML">
<property file="project.properties" />
<target name="massloadSiteXML">
  <exec executable="massload.cmd" dir="src/site/xml">
    <arg value="-infile" />
    <arg value="site_custom.xml" />
    <arg value="-dbname" />
  </exec>
</target>
</project>
```

```

    <arg value="\${DataBase}" />
    <arg value="-dbuser" />
    <arg value="\${DBUserName}" />
    <arg value="-dbpwd" />
    <arg value="\${DBPassword}" />
    <arg value="-method" />
    <arg value="sqlimport" />
    <arg value="-commitcount" />
    <arg value="1000" />
    <arg value="-maxerror" />
    <arg value="1" />
  </exec>
</target>
</project>

```

Rejar and deploy commands and databeans

This project contains a little more logic than the previous two. We are deploying both CustomCommands and CustomDatabeans in the same project. The build file make it possible to only have one of the two. It tests if the JAR files exists and only executes the target if the corresponding file is there.

To rebuild the JAR files we use the Ant tasks unjar and jar. To deploy the JAR files to the correct WebSphere Commerce directories we use the copy task. The mkdir and delete tasks are also used.

The properties WAS_HOME, WCS_HOME, InstanceName, CustomCommandsJar and CustomDatabeansJar should all be added to the project.properties file.

Example 7-5 deployJars - carriage return in XML attributes is used for readability only.

```

<project default="init">
  <property file="project.properties" />
  <target name="init">
    <available file="\${ProjectDir}/src/site/jar/\${CustomCommandsJar}"
      property="CustomCommandsJar.present" />
    <available file="\${ProjectDir}/src/site/jar/\${CustomDatabeansJar}"
      property="CustomDatabeansJar.present" />
    <antcall target="deployCommands"/>
    <antcall target="deployDatabeans"/>
  </target>
  <target name="deployCommands" if="CustomCommandsJar.present">
    <delete file="\${ProjectDir}/dist/site/jar/\${CustomCommandsJar}" />
    <mkdir dir="\${ProjectDir}/dist/site/jar" />
    <delete dir="\${ProjectDir}/build/site/jar" />
    <mkdir dir="\${ProjectDir}/build/site/jar" />
    <unjar src="\${ProjectDir}/src/site/jar/\${CustomCommandsJar}"

```

```

        dest="${ProjectDir}/build/site/jar" />
<jar jarfile="${ProjectDir}/dist/site/jar/${CustomCommandsJar}"
    basedir="${ProjectDir}/build/site/jar" />
<copy file="${ProjectDir}/dist/site/jar/${CustomCommandsJar}"
    todir="${WAS_HOME}/installedApps/
        WC_Enterprise_App_${InstanceName}.ear/wcstores.war/WEB-INF/lib"
    overwrite="yes"/>
</target>
<target name="deployDatabeans" if="CustomDatabeansJar.present">
    <delete file="${ProjectDir}/dist/site/jar/${CustomDatabeansJar}"/>
    <mkdir dir="${ProjectDir}/dist/site/jar" />
    <delete dir="${ProjectDir}/build/site/jar" />
    <mkdir dir="${ProjectDir}/build/site/jar" />
    <unjar src="${ProjectDir}/src/site/jar/${CustomDatabeansJar}"
        dest="${ProjectDir}/build/site/jar" />
    <jar jarfile="${ProjectDir}/dist/site/jar/${CustomDatabeansJar}"
        basedir="${ProjectDir}/build/site/jar" />
    <copy file="${ProjectDir}/dist/site/jar/${CustomDatabeansJar}"
        todir="${WAS_HOME}/installedApps/
            WC_Enterprise_App_${InstanceName}.ear/wcstores.war/WEB-INF/lib"
        overwrite="yes"/>
</target>
</project>

```

Deploy new Enterprise JavaBeans

This project only applies to new Enterprise JavaBeans and not modifications to existing WebSphere Commerce Enterprise JavaBeans. We do not recommend to modify the existing WebSphere Commerce Enterprise JavaBeans in any way.

The `deployEJBs.bat` in Example 7-6 will take care of some of the work needed to be done to deploy custom made Enterprise JavaBeans.

Example 7-6 deployEJBs

```

<project default="deployEJBs">
<property file="project.properties" />
<target name="deployEJBs">
    <mkdir dir="${ProjectDir}/build/site" />
    <mkdir dir="${ProjectDir}/build/ejbjar" />
    <delete dir="${ProjectDir}/build/ejbjar/DT_temp" />
    <mkdir dir="${ProjectDir}/build/ejbjar/DT_temp" />
    <delete file="${ProjectDir}/dist/site/ejbjar/CustomEJBDeployed.jar" />
    <exec executable="ejbdeploy.bat" dir="${ProjectDir}/src/site/ejbjar">
        <arg value="${ProjectDir}/src/site/ejbjar/CustomEJBDeployed.jar" />
        <arg value="${ProjectDir}/build/site/ejbjar/DT_temp" />
        <arg value="${ProjectDir}/dist/site/ejbjar/CustomEJBDeployed.jar" />
        <arg value="-nowarn" />
        <arg value="-keep" />
    </exec>
</target>
</project>

```

```

        <arg value="-35" />
    </exec>

    <exec executable="modifyIsolationLevel.bat" dir="{WCS_HOME}/bin" >
        <arg value="-jarFile" />
        <arg value="{ProjectDir}/dist/site/ejbjar/CustomEJBDeployed.jar" />
        <arg value="-logFile" />
        <arg value="{WCS_HOME}/logs/modifyIsolationLevel.log" />
        <arg value="-dbType" />
        <arg value="DB2" />
    </exec>
</target>
</project>

```

As described in the use case the user will now have to take over to finish the process of deploying the new Enterprise JavaBeans. Due to the relatively long process of deploying new Enterprise JavaBeans it is important to be very thorough in the store design phase. This will enable to development team to build and deploy if not all then most of the required Enterprise JavaBeans in the beginning of the development process.

Attention: Please refer to the readme file on the WebSphere Commerce installation CD for information regarding exporting the EAR configuration information from WebSphere Application Server.

Tip: If you are having problems with the web modules starting to load *before* all the EJB modules have finished loading, you should use the Application Assembly Tool to remove and read the war files from the EAR file.

Building and deploying the Store archive.

Together with the Commands and Databeans target this target will probably be executed more than once a day during the development process. Having a test box with a recent update of the store is essential to the developers and testers. The project manager will also be able to follow the development process more closely.

deploySar.bat in Example 7-7 packs up the SAR file and executes the store publishing utility.

Example 7-7 deploySAR

```

<project default="deploySAR">
<property file="project.properties" />
<target name="deploySAR">

```

```

<delete dir="${ProjectDir}/build/sar" />
<mkdir dir="${ProjectDir}/build/sar"/>
<zip zipfile="${ProjectDir}/build/sar/webapp.zip"
  basedir="${ProjectDir}/src/sar/webapp" update="false"
/>
<zip zipfile="${ProjectDir}/build/sar/properties.zip"
  basedir="${ProjectDir}/src/sar/properties" update="false"
/>
<copy todir="${ProjectDir}/build/sar/data">
  <fileset dir="${ProjectDir}/src/sar/data" />
</copy>
<copy todir="${ProjectDir}/build/sar/SAR-INF">
  <fileset dir="${ProjectDir}/src/sar/SAR-INF" />
</copy>
<delete dir="${ProjectDir}/dist/sar" />
<mkdir dir="${ProjectDir}/dist/sar" />
<zip zipfile="${ProjectDir}/dist/sar/${StoreName}.sar"
  basedir="${ProjectDir}/build/sar"
/>

<copy file="${ProjectDir}/dist/sar/${StoreName}.sar"
  todir="${WCS_HOME}/instances/${InstanceName}/sar" overwrite="yes"
/>

<exec executable="publishstore.bat" dir="${ProjectDir}">
  <arg value="${StoreName}.sar" />
  <arg value="${HostName}" />
  <arg value="${UserName}" />
  <arg value="${Password}" />
  <arg value="${SARPublishMode}" />
  <arg value="${SARXML}" />
  <arg value="&quot;${WAS_HOME}/installedApps/
    WC_Enterprise_App_${InstanceName}.ear/
    wcstores.war=webapp.zip,${WAS_HOME}/installedApps/
    WC_Enterprise_App_${InstanceName}.ear/wcstores.war/
    WEB-INF/classes=properties.zip&quot;;"
  />
</exec>
</target>
</project>

```

Line breaks inside attributes are used for display purposes only.

Important: When republishing an existing store archive, you must ensure that the store directory is not locked by another process. When deploying the store archive the publish utility must be able to rename the existing store archive to create a backup before republishing.

7.3.5 Conclusion

Ant can be used in a number of ways to simplify repeating build processes. Not only does Ant simplify the process but it also eliminates human error during deployment. A lot of things can go wrong if the ones responsible for deployment have to do everything manually while reading an installation guide. Ant will do the same thing every time no more no less.

We have only showed some very simple features of Ant. But it can do so much more. There is in fact a plugin that allows Ant to connect to a VisualAge for Java repository. This would eliminate the process of having to export the code from VisualAge for Java manually.

When the WebSphere Commerce store development moves to WebSphere Studio Application Developer things will be even simpler.



Part 1

Customization examples

In this part we describe customization examples that extend the functionality of WebSphere Commerce.



Examples overview

This chapter serves as an outline of the second part of the redbook. We introduce our sample stores, examine how they work and explain how to take advantage of the extended functionality provided. To provide a simple user's guide, we walk you through the interfaces of the stores, pointing out the customized features. This chapter is best used in conjunction with the chapters that follow, referring to both, while working with the examples.

The approach we've taken is to extend the sample stores that are provided with WebSphere Commerce Business Edition V5.4 with a focus on a particular area of customization. This should put you on familiar footing with respect to usability while showing you something new about the product. Information about the basic functionality of the out-of-the-box samples stores can be found at:

http://www.ibm.com/software/webservers/commerce/wc_be/1it-tech-general.html

8.1 Example stores

WebSphere Commerce provides developers with several out-of-the-box sample stores on which they can base their own stores. These sample stores each showcase various aspects of the WebSphere Commerce product.

The out-of-the-box samples that we use to create our stores in the example chapters that follow are WebFashion and ToolTech. WebFashion employs the business-to-consumer store model. It's built-in features include discounts, bundles and wish lists. ToolTech, on the other hand, is a business-to-business store. It focuses on items such as contract-based purchasing, requisition lists and request for quote (RFQ) creation. Detailed information about the stores and how to install them can be found in the online help files for WebSphere Commerce.

8.2 Orders

In this set of examples, found in Chapter 9, “Orders” on page 233, we focus on how orders can be processed in WebSphere Commerce Business Edition.

Using the ToolTech sample store as a base, the first example shows an implementation of an interface between the commerce system and a back-end system. The store will pass an order to a CICS program simulating an order being sent to a back-end system.

The second example in this chapter, also an extension of the ToolTech sample store, features a detailed explanation of the quick order function. Quick order exists as a built-in function in WebSphere Commerce Business Edition Version 5.4; we show by example how to use it in a store.

8.2.1 CICS order transaction

This example is centered around interaction with a back-end system. For the purpose of this exercise, we are using a CICS system as our back-end. This CICS program could represent an ERP or CRM system in the enterprise. In fact, it is quite common for a business's commerce system to have to interface with its other resources. As companies work to incorporate the various functions of the resources in their environments, integration becomes critically important.

CICS, or Customer Information Control System, is an online transaction processing program that provides tools for building customer transaction applications, traditionally on large enterprise mainframes. Using the CICS application programming interface, developers can write programs that

communicate with users and read from or write to records in a database in a transactional manner. CICS provides resource management leading to improved load-balancing and highly scalable systems. CICS can also ensure that transactions are completed and, if they are not, it can undo partly completed transactions to maintain data integrity.

The CICS Transaction Gateway provides secure, flexible access from Web applications to business-critical applications running on a CICS server on the back-end using standard Internet protocols. We use the connectors provided by CICS Transaction Gateway to facilitate the interface between the commerce system and the CICS server.

In the example we show how to create a new command to process an order by sending it to the CICS system via a CICS Transaction Gateway. This CICS system can represent any application on a back-end system.

Implemented in such a way that it is invisible to the user, this command is called when a user places an order. An order reference number and the total cost of the order is packaged and sent to the CICS system via the CICS Transaction Gateway.

The CICS application responds with a text message indicating the transaction's success or failure. The CICS response is stored in the database. This response can be used to perform an action on the commerce system or notify the user of a problem. A developer or administrator can confirm that the transaction succeeded by checking WebSphere Commerce's default message log or the ORCOMMENT table in the database where the CICS transaction information is stored.

8.2.2 Quick order function

Quick order is a very useful tool for a store which has a lot of repeat orders. It provides an efficient way to add items to an order by allowing the customer to input product identifiers and amounts and immediately adding those products to the shopping cart.

In this example, we examine the implementation and operation of the built-in quick order function of the ToolTech store. As a user browses the store, there is a section on the left hand navigation bar labelled Quick Order. The Quick Order section contains an **Enter More Items** link which leads to the quick order screen. From this screen the user can input the product information (SKUs in this case) and the amount they want to order. Clicking **Order** from this screen adds the items directly to the shopping cart which is then displayed to the user. Figure 8-1 shows the quick order page.

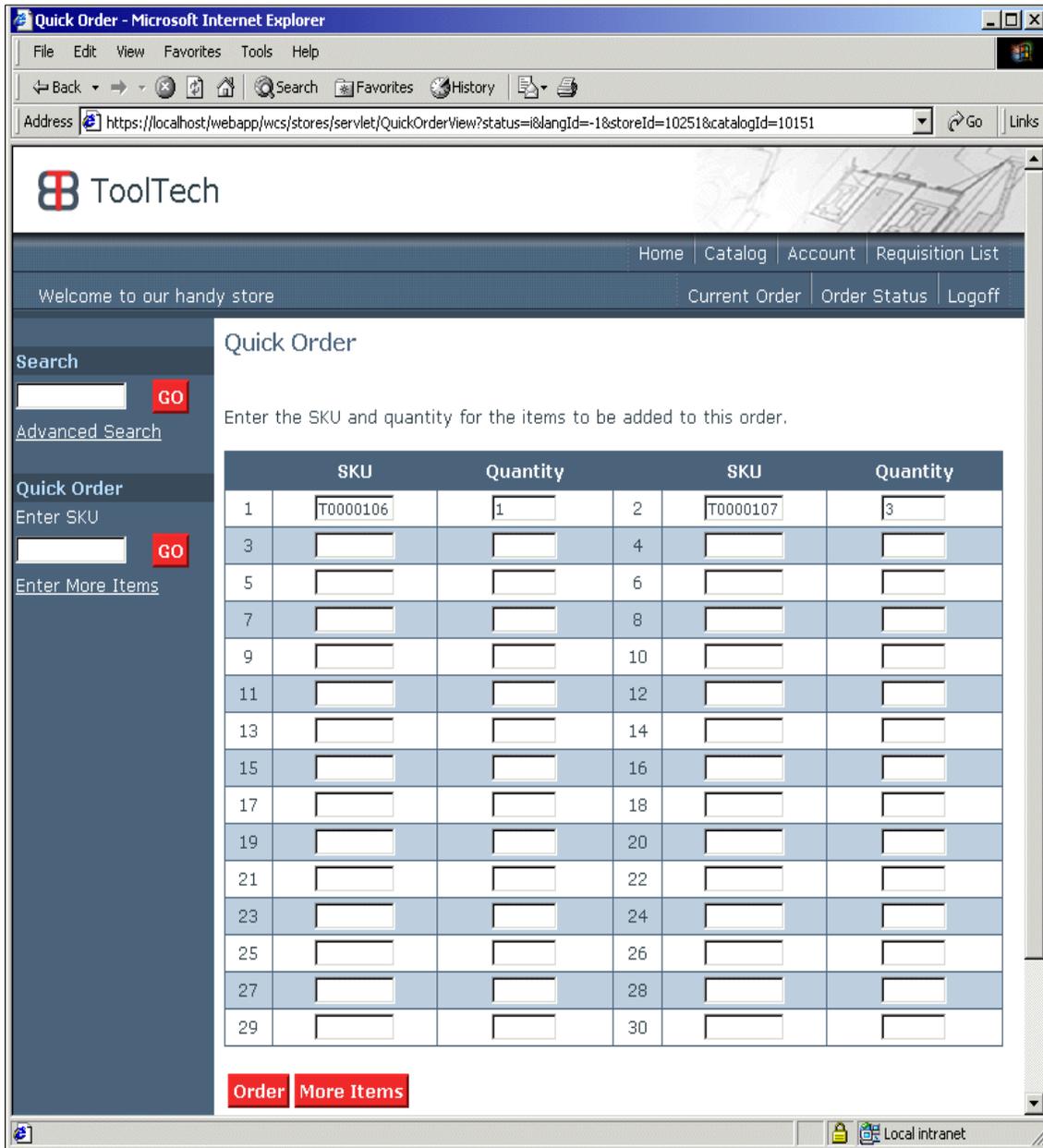


Figure 8-1 Quick order Web page from ToolTech

8.3 Shipping and taxes

The set of examples in Chapter 10, “Shipping and taxes” on page 259 are centered around the processes surrounding shipping and taxation. These topics figure prominently in a global business and a store needs to be aware of the various standards in each country and know how to accommodate them.

We first examine the implementation of the built-in weight-based shipping function, reviewing the associated tables, commands and files. Using the InFashion sample store, we also consider the implementation of a product display that includes taxes in the pricing. Finally, we again extend the InFashion sample store to explore discounting prices.

8.3.1 Shipping by weight

In this example, we focus on how a store ships products. Three things must be defined in the commerce database in order for a store to ship orders:

Shipping mode	A combination of the company that ships the product to the customer and the type of shipping that will be used (e.g., overnight service, 2-day service, ground service)
Shipping arrangement	The agreement between the store and a fulfillment center outlining the conditions that will be applied for shipments for the store, including such things as shipping zone restrictions and shipping methods
Shipping charge	The method for calculating the charge for shipping orders to customers

There are several types of shipping charges: a flat fee for all orders, a flat fee for each item in an order, a percentage of the cost of the order, a weight-based charge or a quantity-based charge. For this example, we examine the implementation of the weight-based charge.

Figure 8-2 shows the weight rates screen in Store Services for the ToolTech sample store. You can see from this figure the various charges that are associated with the weight ranges that have been configured for the store. Note also that the shipping mode is displayed as each mode can have different charges associated with it.

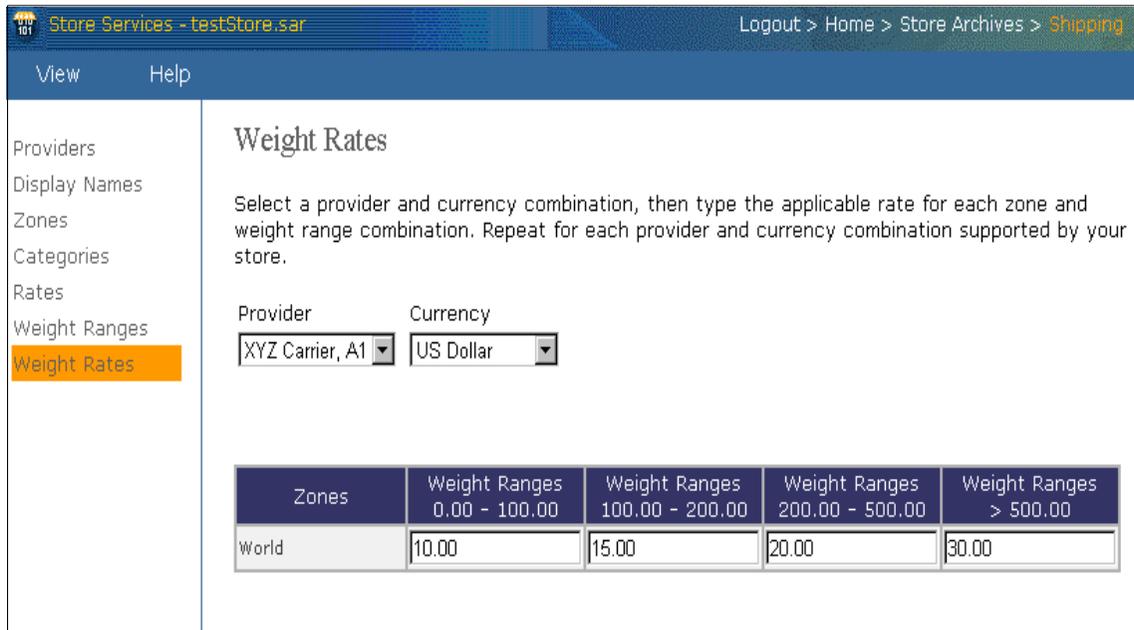


Figure 8-2 Shipping by weight settings in WebSphere Commerce Store Services

8.3.2 Display prices with tax

Tax laws and practices vary greatly from country to country and, often, from one area to another within a country. With the rapid globalization the Internet has enabled, awareness of many taxation customs has taken on increased importance. This section looks at the practice of including taxes in the product price that is displayed to customers.

We begin by explaining how taxes are calculated and applied. We also discuss the assets associated with taxation in the WebSphere Commerce data model and introduce tax-related WebSphere Commerce terms.

Extending the InFashion sample store, we implement the display of prices with tax inclusive. The integration of the example presented can be seamless with the appropriate price shown based on the store's needs.

8.3.3 Discounts

Discounts are a frequently used marketing tool that may be used to make a product more attractive to customers. In WebSphere Commerce, discounts may be offered as percentages or fixed amounts and may apply to a single product or an entire order. As in the brick-and-mortar world, discounts incent customers to visit the store and purchase certain products.

An administrator can get very granular with the application of discounts in the commerce system. WebSphere Commerce lets one control:

- ▶ Time period during which the discount is in effect
- ▶ User subset to which the discount pertains
- ▶ Scope of the product catalog ythe discount covers
- ▶ Method by which the discount is applied

A store can also apply discounts to a range of prices resulting in a graduated discount scale. Figure 8-3 shows the definition of ranges through WebSphere Commerce Accelerator.

The screenshot shows the 'Ranges' configuration page in WebSphere Commerce Accelerator. The page title is 'Ranges' and it includes a navigation menu with 'Store', 'Marketing', 'Merchandise', 'Operations', and 'Help'. The page content includes a 'General' section with a description: 'Discount ranges are based on the price. To specify discount ranges, click **Add** and enter the **"Range Start"** value and the **"Discount"** value. The **"Range End"** is automatically calculated.' Below the description is a table with the following data:

	Range start (USD)	Range end (USD)	Discount (USD)
<input type="checkbox"/>	0.00	99.99	0.00
<input type="checkbox"/>	100.00	499.99	10.00
<input type="checkbox"/>	500.00	999.99	20.00
<input type="checkbox"/>	1,000.00	and up	30.00

To the right of the table are three buttons: 'Add', 'Modify', and 'Remove'.

Figure 8-3 Defining discount ranges in WebSphere Commerce Accelerator

The example, an extension of the InFashion sample store, shows the application of a per product discount for the entire product catalog for registered users. A customer browses the site as he or she normally would; no special customer action is necessary. The discount is applied to the order automatically and is displayed to the customer in the order summary.

8.4 Messaging customization

MQSeries allows stores to connect disparate systems together resulting in a more seamless integration of your WebSphere Commerce environment with any other systems you may have. With over thirty five platforms currently supported by MQSeries, you can connect Windows platforms to OS/390, iSeries to Solaris, Linux to MVS and any of a number of other combinations. MQSeries assured one-time delivery makes it a valuable tool for e-business.

8.4.1 Inbound MQSeries - product creation

Typically, there are two ways to create new products in WebSphere Commerce: the MassLoader tool and WebSphere Commerce Accelerator. Our goal in Chapter 11, “Messaging customization” on page 291 is to introduce a third method: create the product on a back-end system and automatically push the update to the WebSphere Commerce system using MQSeries.

In this example, an administrator creates a new product or modifies an existing product on a back-end system. The back-end system creates a well-defined XML file describing the product and its attributes and puts that file on an MQSeries queue for delivery to the commerce system.

The MQ adapter on the WebSphere Commerce system receives the message from the queue and parses the data. Using the product definition, WebSphere Commerce creates or modifies the product in the store’s database by calling the MQProductCreate command as shown in the example. Users are then immediately able to see the new product or product changes as long as the appropriate cache triggers are set.

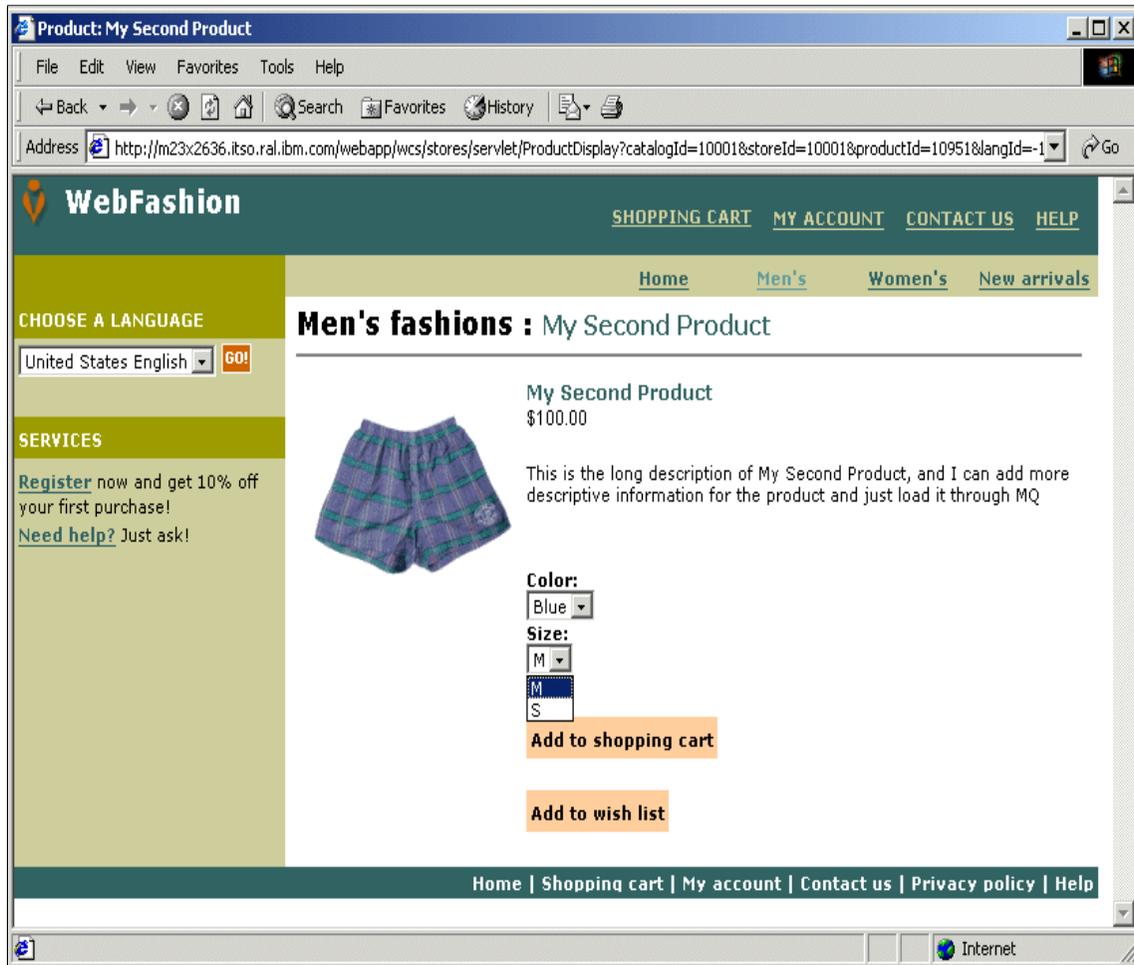


Figure 8-4 Product created using MQProductCreate

Figure 8-4 shows the display of a newly created product that has been defined using the MQProductCreate command. We use the WebFashion sample store to show the implementation of this example, but, as with many of the other examples we present, it is sufficiently abstract to be applied to any store.

8.5 B2B features

Business-to-business (B2B) transactions have received an enormous amount of attention recently and with good reason. Web sites that support B2B transactions can increase a company's productivity while simultaneously driving down the costs related with maintaining a partner relationship. B2B figures prominently in WebSphere Commerce Business Edition Version 5.4 with components aimed specifically at the needs of this market including:

- ▶ The ability to process orders using purchase order numbers included in the order.
- ▶ Support of requisition lists to facilitate frequent, repeated orders.
- ▶ The restriction of catalog browsing based on contract terms.
- ▶ Login pages personalized to individual organizations.
- ▶ Support for creation of requests for quotation (RFQs).

The ToolTech sample store packaged with WebSphere Commerce Business Edition highlights many B2B features. In this example, we extend the basic functionality of the ToolTech store to enable the restriction of catalog viewing and roles-based login. We also implement an amount-based approval method wherein any transaction that is over a certain price will need to be approved by a second party. In this way, we show a more natural, buyer-oriented flow for a B2B site.

8.5.1 Role-based display

A role in WebSphere Commerce defines the type of access a user has and the specific resources that user can modify. While the Site Administrator is the only role that has the authority to create, assign or unassign roles to or from all users and organizational entities, Buyer Administrators and Seller Administrators may assign or unassign roles for the organizational entity to which they belong and to organizational entities below that. Buyer Administrators and Seller Administrators may also assign or unassign roles to users within those entities and to themselves. A user may be assigned more than one role, but an organizational entity must be assigned a given role in order for a user within that entity to be assigned that role. Figure 8-5 shows a portion of the roles management section of the WebSphere Commerce administration console.

Site Administration Console Logout > Home > Roles

Access Management Approvals Security Performance Configuration

Roles

Page Number

24 items « [First](#) | 1 of 2 | [Next](#) » | [Last](#) »

<input type="checkbox"/>	Name	Description	New
<input type="checkbox"/>	Account Representative		
<input type="checkbox"/>	Buyer (buy-side)		
<input type="checkbox"/>	Buyer (sell-side)		
<input type="checkbox"/>	Buyer Administrator		
<input type="checkbox"/>	Buyer Approver		
<input type="checkbox"/>	Category Manager		
<input type="checkbox"/>	Customer Service Representative		
<input type="checkbox"/>	Customer Service Supervisor		
<input type="checkbox"/>	Logistics Manager		
<input type="checkbox"/>	Marketing Manager		
<input type="checkbox"/>	Operations Manager		
<input type="checkbox"/>	Pick Packer		
<input type="checkbox"/>	Procurement Buyer		
<input type="checkbox"/>	Procurement Buyer Administrator		
<input type="checkbox"/>	Procurement Manager		

Figure 8-5 Roles management screen in WebSphere Commerce

In our example, we extend the ToolTech sample store to show that the role assigned to a user can be used to tailor pages. A user assigned to a specific role can view one page while another user with a different role may view another page altogether.

We present the method used to retrieve a user's roles, then use that information to display a home page that changes dependent upon what roles the user has. If the user is a Buyer Approver or Buyer Administrator, he or she will be served a page that is different from what other users see. There will be additional links on the Buyer Approver and Buyer Administrator's home page for tasks that are specific to them.

This can be a powerful tool for personalization, as stores can easily and dynamically cater to a multitude of different customers by creating roles as necessary and targeting the users associated with those roles.

8.5.2 Amount-based order approval

In a business-to-business setting, it is often useful to have a person or group of people in a purchasing company responsible for approving orders submitted by the company's buyers. To accommodate this, WebSphere Commerce Business Edition allows store owners to quickly and easily enable this feature at a contract level.

A price threshold can be established for a contract; in the case of our example, that threshold is \$250. When the threshold is met or exceeded, the order, upon submission by a buyer, is placed into pending state.

A user who has been set up to provide order approval will be informed of the order and take the appropriate action by either approving the order, which then flows through the system as it normally would, or rejecting the order. If the order is rejected, then some tailored post-processing may occur to inform the buyer that the order has been rejected

The approval or rejection of orders in WebSphere Commerce is a seamless part of the store. The approver logs into the store, proceeds to the approval page and takes an action, either approving or rejecting the orders. The user tasked with approval must exist in the commerce system; otherwise buyers may not be able to submit orders that need to be approved.

8.5.3 Contract-based shopping

In WebSphere Commerce, a contract is defined as an agreement outlining the terms and conditions that apply to a transaction. All customers shop under a contract. Pricing is set at the contract level resulting in a situation wherein the same product may be sold at different prices depending upon the contract being used.

By default, WebSphere Commerce's out-of-the-box business-to-business sample store allows users to select a contract for each product as they shop. If a user is entitled to use multiple contracts, he or she will see the prices associated with each of the contracts. This flow is shown in Figure 8-6 where the user is presented with three contracts that may be used.

The screenshot shows the ToolTech e-commerce website. The header includes the ToolTech logo, a navigation menu with 'Home', 'Catalog', 'Account', and 'Requisition List', and a user status area with 'Welcome to our handy store', 'Current Order', 'Order Status', and 'Logout'. The main content area is titled 'Item Display' and features a product image of a blue air-powered nailer. The product details include the name 'Air Powered Nailer', a description, and technical specifications: 'Fastener capacity : 60' and 'Operating pressure : 120'. A 'Quantity' input field is set to 1. Below the product information is a table for selecting a contract. The table has two columns: 'Contract' and 'Price'. Three contracts are listed, each with a radio button for selection. The prices are \$328.00, \$262.40, and \$328.00 respectively. An 'Add to Order' button is located below the table. On the left side of the page, there are search and quick order sections.

Search
 GO
[Advanced Search](#)

Quick Order
 Enter SKU
 GO
[Enter More Items](#)

Item Display

Air Powered Nailer
 The industrial power of this air powered nailer is shown by it's 60-80 fastener capacity and 70-120 psi operating pressure. Easy grip handle reduces wrist strain and allows for greater accuracy.

SKU: T00001251
Item ID: 10233
Manufacturer: Stamina Tools
Manufacturer part number: Stamina-Tools-1251

Fastener capacity : 60
Operating pressure : 120

Quantity

Select a contract for this order in the table and click on the button below to add this item to your order.

	Contract	Price
<input type="radio"/>	testStore Contract number 1234 [10003]	\$328.00
<input type="radio"/>	testStore Contract number 2345 [10004]	\$262.40
<input type="radio"/>	Tao Inc [10052]	\$328.00

Add to Order

Figure 8-6 Shopping with contracts

It may be desirable, however, to associate a particular order to a contract at the beginning of the shopping flow to allow for a faster flow through the site. This is the approach we examine in our example.

A user of the store is prompted for the contract to use as soon as he or she logs in to the system. Everything the user purchases as a part of that order will be associated with that contract.

Contract-based shopping also makes it easy to restrict a user's view of the product catalog. Items that cannot be purchased under a specific contract need not be shown at all.

8.6 Product entry and display

This set of examples shows how to implement and extend WebSphere Commerce functionality involving the display of the catalog. We examine the product comparison function with a goal of customizing it so that it will return like products from the catalog when prompted by the user. We review the implementation of bundles by extending the WebFashion sample store. We also enable the display of multiple currencies simultaneously.

8.6.1 Product comparison

Product comparison is one of a set of several shopping metaphors defined in WebSphere Commerce. Shopping metaphors provide customers with several ways to find the products they want. The shopping metaphors available in WebSphere Commerce Business Edition Version 5.4 are:

- ▶ Product comparison
- ▶ Product exploration
- ▶ Sales assistance

A store administrator applies the shopping metaphor functions to the store by using the Product Advisor utility. The Product Advisor helps to organize and present the online product catalog along the lines of a given shopping metaphor.

The product comparison function allows users to quickly and easily compare and contrast the features of a set of products side by side. In this example, we show how to enable the product comparison metaphor and tailor the behavior and display of the function.

8.6.2 Enabling bundles

Bundles are a marketing tool used by product and marketing managers to provide a one-click purchasing ability for related items. A bundle can have many items and must have at least one stock keeping unit (SKU). When a bundle is selected for addition to a customer's order, it is decomposed into its orderable components. These components are then added to the order and their prices aggregated to compute the total price.

Once the components are a part of the order, a customer may change the number of each item that is being ordered or deselect specific items. The price is recalculated based on what is added or removed as would normally occur.

As an example of a bundle, consider a computer sold at an online store. The computer system may be bundled with a specific type of monitor, DVD player and extra hard drives. Once these items are added to the customer's cart, the customer may remove the items he or she does not want. This method creates the opportunity for a great amount of cross-selling.

This example uses the WebFashion sample store to walk through the process of defining a new product via Product Manager and the creation of a bundle. We review the assets associated with bundles and how they can be effectively used as a marketing method.

8.6.3 Display of multiple currencies

Currency display, like shipping and taxation, is a topic that has become increasingly important in recent years due to the explosion of globalization. Stores need to be able to operate in all of the currencies that their audience is most comfortable using or risk losing customers.

Displaying multiple currencies simultaneously may be useful in cases such as the euro changeover. As countries continue the migration from an older currency to the euro, many products are displayed using both methods to make the transition easier on the customer.

In this example, the WebFashion store is altered to allow for the display of more than one currency. We continue to use the example of the euro to highlight the benefit that this brings to stores. The example can be extended to allow for displaying a number of different currencies as the need arises.



Orders

This chapter provides an example on how to implement a CICS transaction call within the processing of an order and it will provide some input on how to create a quick order functionality in your e-commerce shop.

This chapter contains the following examples:

- ▶ CICS order transaction
- ▶ Quick Order

9.1 CICS order transaction

CICS (Customer Information Control System) is an application server that handles online transaction processing and it is being used by many thousands of organizations all over the world to take care of their transaction processing needs. CICS will ensure the integrity of your data during the transaction, run efficiently, access protected resources and then terminate when completed.

To get more information about the CICS transaction system, refer to:

<http://www.ibm.com/cics>

This order example will create a task command that will, upon order processing, do a CICS transaction. The example is divided into the following sections:

- ▶ Functional requirements
- ▶ Use cases
- ▶ Design
- ▶ Pre-requisites
- ▶ Create the Task Command
- ▶ Register the command in WebSphere Commerce
- ▶ Deploy the code

9.1.1 Functional requirements

A brief description of the functional requirements is that we are required to develop a module for passing specific order information to a back-end CICS program, which resides on a S/390 server, when the order is being placed.

The following two sections explain the input to the CICS program, the output from the CICS program and what the module is expected to accomplish. The sections are based on information provided at the time the functionality was requested.

Input to the CICS program

The input to the CICS program is to be a comma-separated list, containing the order reference number and the total price of the order. For example 12345,312.54 where 12345 is the order reference number and 312.54 is the total price of the order. The input to the CICS program is to be provided by the External Call Interface (ECI).

An ECI call allows non-CICS applications to call a CICS program on a CICS server. The information between the CICS program and the Java command is exchanged by the means of a COMMAREA.

Note: A COMMAREA (communications area) is a memory area used to transfer data between two programs.

The input to the CICS program is to be a 27 bytes input. Excess byte locations are to contain the byte value 0 if the comma-separated list length is less than 27 bytes.

Output from the CICS program

The CICS program will acknowledge the input information by responding with the application ID, the date and time of the transaction. A sufficient validation of the response from the CICS program is to check that the response message contains two colons.

The response message from the CICS program needs to be stored for future reference.

9.1.2 Use cases

The system use cases of the CICS implementation are simple and are described below.

Basic flow

This is the basic system flow of the CICS transaction:

1. The user adds items to the shopping cart and does a checkout to place the order.
2. When the order is being placed, that is when the OrderProcess command is being executed; the module creates the order message, passes it to the CICS program, retrieves the response, validates it and saves it in the database and in the log file.
3. The order complete page is displayed.

Exception flow

To be able to trace when and why CICS responds with invalid information, an exception flow is necessary to explain the output of that scenario.

1. The user adds items to the shopping cart and does a checkout to place the order.
2. When the order is being placed, that is when the OrderProcess command is being executed, the module creates the order message, pass it to the CICS program, retrieves the response, validates it and the validation fails. The CICS response information is saved in the log file.

3. A rollback occurs and an exception is thrown to a generic page.

9.1.3 Design

When composing the design of the solution, we need to review the input, the tools we can use, and the requested output of the implementation. The task command that we will develop needs to connect to a CICS server, send some information and get an answer back from the server.

To be able to connect to the CICS server, we need to implement connectivity to the CICS Transaction Gateway that provides the choice of three interfaces for communication with CICS. The CICS Transaction Gateway software contains client and server components that will allow a Java application to invoke services in a CICS region and the interfaces options are:

- ▶ Common Connector Framework API
- ▶ The base classes of CICS Transaction Gateway
- ▶ J2EE Common Client Interface

The J2EE Common Client Interface has replaced the Common Connector Framework as a strategic way of using a common standard to connect to Enterprise Information Systems like CICS, hence we choose to use the J2EE Common Client Interface. As the interface to the CICS program is ECI, we will have to use the ECI resource adapter to do this. We could chose to use the base classes of the CICS Transaction Gateway to implement the functionality, but they require a very good understanding of CICS.

The CICS Transaction Gateway application will reside on a Windows 2000 Server and will act as a server daemon for the Java application (the task command) and as a client daemon for the CICS program on the CICS server.

We decide to develop the application to run in a non-managed environment, which means that the command will need to manage the connection to the server and the transmittal of the transaction as well as security issues. We are choosing a non-managed environment to be able to provide a more interesting example, as it will have to discuss the communication process in more detail.

Note: In a managed environment the command would access the CICS resource adapter through an application server such as WebSphere Application Server.

Refer to the redbook *Java Connectors for CICS: Featuring the J2EE Connector Architecture*, SG24-6401, to get an extensive in-depth guidance to the options available in implementing a CICS connection with Java.

A summary of the transaction design is shown in Table 9-1.

Table 9-1 Transaction design

What	Value
Interface to be used	J2EE Common Client Interface - The ECI resource adapter
Managed environment	No

Properties

To be able to externalize the communication to the CICS Transaction Gateway we need to store the communication properties in such a way that the information is unique and can easily be changed or updated. We decide to use the STADDRESS table to keep the CICS Transaction Gateway properties as it provides us with the necessary columns and is related to the owner of the shop (MEMBER ID). We use the name CTG-PROP as a value in the NICKNAME column to identify the correct information when doing a lookup to the table.

Another option is to utilize a property file to keep the configuration information. A property file would, performance wise, be a more appropriate choice, as no lookups to the database would be necessary. Nevertheless we chose in this example to use the database as it does not require an additional file added to the system.

We design a row in the STADDRESS table to keep the information as specified in Table 9-2.

Table 9-2 Property settings

Column	Value
STADDRESS_ID	The unique key in the STADDRESS table.
MEMBER_ID	The owner ID for this entry, which is the member ID of the shop.
ADDRESS1	The URL to the CICS Transaction Gateway
ADDRESS2	The CICS region.
ADDRESS3	The service to be called on the CICS server.
BUSINESSTITLE	The title of the store
NICKNAME	CTG-PROP

CICS transaction result

As a requirement, the response from the CICS transaction needs to be stored in the database as well as being written to a log file. The information in the log file is necessary to handle the possibility that the command fails after the CICS transaction has been executed. The information in the log file would then still be presenting the CICS transaction information but the information added to the database would never be committed and consequently get lost. We will use the default message log of WebSphere Commerce to output the transaction information.

To store the CICS transaction information, we should use an order related table and the ORCOMMENT table suits our purpose of storing a string of information. The information will be stored as specified in Table 9-3.

Table 9-3 ORCOMMENT table

Column	Value
ORCOMMENT_ID	Generated unique key
ORDERS_ID	The order reference of the related order to the CICS transaction
COMMENTS	The CICS transaction result

9.1.4 Pre-requisites

In order to be able to develop the task command and to test it, some pre-requisites needs to be fulfilled which we cover in this section.

Knowledge

The reader needs basic understanding on what a CICS Transaction Gateway is and how it works. Refer to the CICS Transaction Gateway documentation that comes with the CICS Transaction Gateway installation CD.

Test environment

In order to be able to test the task command during development, we need a test environment. In this redbook we used the following test environment:

- ▶ We had a Windows server running CICS Transaction Gateway V4.0.1. The URL to the CICS Transaction Gateway was `tcp://gunner.almaden.ibm.com` and the CICS Transaction Gateway listened on port 2006.

Note: Port 2006 is the default port of a CICS Transaction Gateway set up.

- ▶ We had a CICS program named ECIPROGX to call on an S/390 server, which is identified by the CICS Transaction Gateway as the CICS server (region) SCSCPAA6.
- ▶ The CICS program is the one that will be used in production and it does not require a username and password.

Development environment

The instructions in this chapter will assume that you have a properly installed WebSphere Commerce Studio environment with a WebSphere Commerce configured version of VisualAge for Java, Enterprise Edition.

To be able to run the task command we develop, we need to configure VisualAge for Java. To run a J2EE ECI CICS application in a non-managed environment, as our task command will, we need to add the following JAR packages to the runtime classpath:

- ▶ cicsj2ee.jar
- ▶ ctgclient.jar
- ▶ ccf2.jar
- ▶ connector.jar
- ▶ screenable.jar

If the Java application were to run outside a J2EE environment, we would have to add the JTA Java extension from Sun to the classpath.

Note: Installations of VisualAge for Java as well as WebSphere Application Server already include the JTA Java extension.

If we were to keep the CICS Transaction Gateway on the same machine as the Java application, we would have to add the ctgserver.jar file to the classpath as well.

The JAR files can be found in a subdirectory called classes from the installation of the CICS Transaction Gateway 4.0.1 and they are also supplied as additional material to this redbook.

Set up VisualAge for Java

To set up VisualAge for Java, we will add the VisualAge for Java features which contains the necessary files. Complete the following steps to set up VisualAge for Java to enable development of the CICS transaction in the task command:

1. Exit VisualAge for Java.

2. Run one of the following setup programs, as appropriate for your system, from the VisualAge for Java Additional Features CD and follow the installation steps:
 - extras \BetaJ2EEConnectors\NT\Setup.exe
 - extras \BetaJ2EEConnectors\W2000\Setup.exe
3. When the installation is completed, start VisualAge for Java.
4. Verify that the workspace owner is the Administrator. If the workspace owner is not the Administrator, from the main menu of VisualAge for Java select **Workspace -> Change Workspace Owner**. Select the Administrator and click **OK**.
5. Delete any of the following projects that may exist in your workbench:
 - Connector CICS
 - IBM Common Connector Framework
 - IBM Enterprise Access Builder Library
 - IBM Enterprise Access Builder Samples
 - IBM Enterprise Access Builder WebSphere Samples
 - IBM Java Record Library
6. On the main menu in VisualAge for Java, select **File -> Quick Start**.
7. When the Quick Start window appears, select **Features** and **Add Feature**. Click **OK**.
8. When the Selection Required window appears, select IBM Java Record Library 3.5.3.5 and J2EE Connector Architecture 1.0 as shown in Figure 9-1. Click **Ok**.

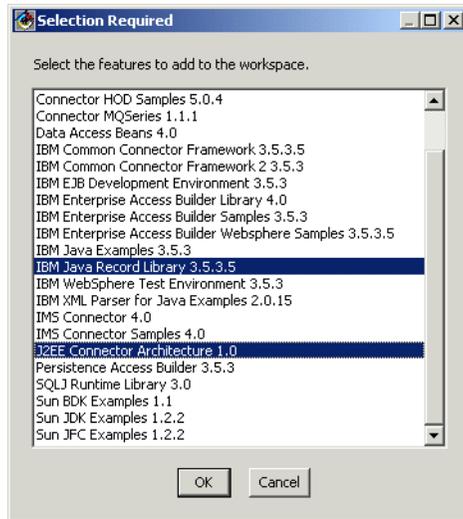


Figure 9-1 Add Features to VisualAge for Java options

9. When the projects are imported, we need to update the J2EE Connector Architecture version from a Proposed Final Draft 2 to Version 1.0. Right-click the **J2EE Connector Architecture** project on the workbench and select **Import**.
10. The Import SmartGuide appears. Select to import a JAR file and click **Next**.
11. Click **Browse** and select the connector.jar file that is found in a subdirectory called classes from the installation of the CICS Transaction Gateway 4.0.1.
 Select all class files to be imported as well as resource files. Verify that the files are to be imported into the J2EE Connector Architecture project and make sure that the **Create new/scratch editions of versioned projects/packages** option is checked. The window will look like Figure 9-2. Click **Finish**.

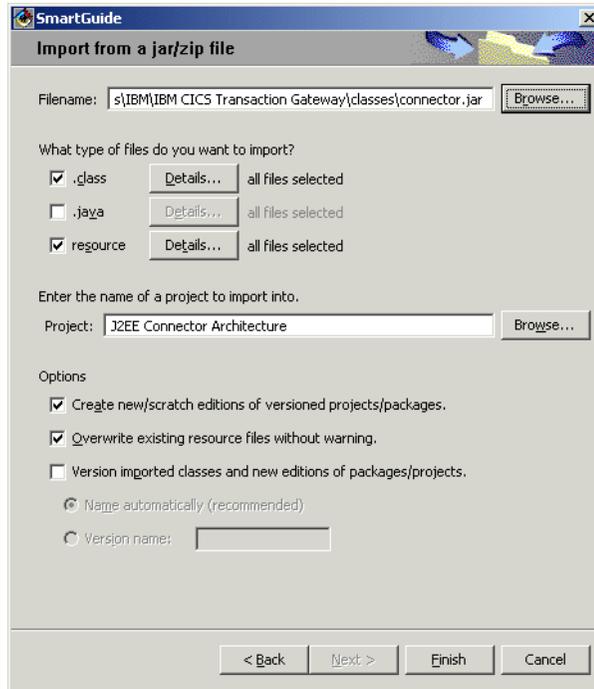


Figure 9-2 Import connector.jar window

12. When all the files have been imported to the J2EE Connector Architecture, it is a good idea to version the project.
13. Now we need to import the CICS Transaction Gateway related classes. From the main menu of VisualAge for Java, select **File -> Import**.
14. Select to import a JAR file. Click **Next**.
15. From the same directory as we got the connector.jar file we select the ctgclient.jar file.
Enter **CICS Connector** as the project name and set the remaining import options as shown in Figure 9-2. Click **Finish**.
16. A window named Modify Palette will appear in which you can modify the bean palette used by the Visual Composition Editor for visually composing applications using Java Beans. We will not do any visual composing, so click **Cancel** to complete the import.
17. Repeat step 13 to 16 to import the following JAR files to the CICS Connector project:
 - cicsj2ee.jar

- screenable.jar
- 18. Version the CICS Connector project.
- 19. Change the workspace owner to the WCS Developer.

9.1.5 Create the Task Command

When placing the order, the user is calling a URL command called `OrderProcess`. This command will at the near end of its process call for a task command named `ExtOrderProcess`. The `ExtOrderProcess` command is by default calling a dummy implementation class that doesn't do any logic and we will replace this dummy implementation class with the one that we will create to be able to perform the CICS transaction.

The instructions in this section will refer a lot to the additional material for this chapter. The only method that we will discuss in detail is the method that does the CICS transaction and all the other methods will only be discussed very briefly.

The code discussed in this chapter can be obtained with the additional material from this redbook. The source code is in the `cicstrans_src.jar` in the `sg246190.zip` file. See Appendix B, "Additional material" on page 409 for details of how to obtain the redbook material.

Create a class in `VisualAge` for Java named `com.ibm.commerce.wc54handbook.commands.DoCicsTransJ2EECmdImpl` that extends `com.ibm.commerce.command.TaskCommandImpl` and implements `com.ibm.commerce.order.commands.ExtOrderProcessCmd`.

The implementation of the `ExtOrderProcessCmd` interface requires that you implement a getter and setter method in the implementation class for the order reference number. When we have created the class and the first necessary methods and fields, the class will look something like Example 9-1.

Example 9-1 DoCicsTransJ2EECmdImpl class

```
public class DoCicsTransJ2EECmdImpl
    extends com.ibm.commerce.command.TaskCommandImpl
    implements com.ibm.commerce.order.commands.ExtOrderProcessCmd {

    public static final String COPYRIGHT =
        com.ibm.commerce.copyright.IBMCopyright.SHORT_COPYRIGHT;
    private java.lang.Long orderId;
    public static final String CLASSNAME = "DoCicsTransJ2EECmdImpl";

    public Long getOrderRn() {
        return orderId;
    }
}
```

```

    }

    public void setOrderRn(Long param1) {
        orderId=param1;
    }
}

```

Main business logic of the command

The first thing to do when writing the task command is to write the `performExecute` to get a good structure of the command. The method is based on the functional requirements and thus is quite easy to implement when we have the design already written for the command.

The necessary actions in the `performExecute` method are:

1. Get the CICS Transaction Gateway properties. We store the CICS Transaction Gateway properties in a Hashtable object.

```

java.util.Hashtable ctgProp = null;
ctgProp = getCTGprop();

```

2. Get the order information that is to be sent to the CICS program and format the information into the CICS input message specified format.

```

String orderData = null;
orderData = getOrderData();

```

3. Do the CICS transaction and get the result string. To do the transaction we pass the CICS Transaction Gateway properties and the string of data to the method.

```

String transRes = doTransaction(ctgProp, orderData);

```

4. Write the CICS program response to the WebSphere Commerce log file. The message should contain the order reference and the response from the CICS program.

```

ECMessageLog.out(
    ECMessage._INF_GENERIC,
    CLASSNAME,
    methodName,
    ECMessageHelper.generateMsgParms(
        "CICS Transaction information for orderId="
        + getOrderRn()
        + ": CICS Trans response='"
        + transRes
        + "'");

```

5. Validate the response from the CICS program.

```

validateResponse(transRes);

```

6. Store the CICS information in the database.

```
createOrComment(transRes);
```

The performExecute method needs to be updated with exception handling and trace statements to ease troubleshooting. Modify the preformExecute method after each of the methods it is calling has been created. The final performExecute method will look something like Example 9-2.

Example 9-2 performExecute method

```
public void performExecute() throws com.ibm.commerce.exception.ECException {
    String methodName = "performExecute";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    //Call the performExecute of the superclass
    super.performExecute();

    /*=====
    Get the CICS Transaction Gateway settings.
    =====*/
    java.util.Hashtable ctgProp = null;
    try {
        ctgProp = getCTGprop();
    } catch (javax.ejb.FinderException fe) {
        //Didn't find the row in the STADDRESS table
        throw new ECSystemException(
            ECTraceIdentifiers.COMPONENT_EXTERN,
            CLASSNAME,
            methodName,
            ECTraceHelper.generateMsgParms(
                "Could not find CTG-properties in STADDRESS table.));
    } catch (Exception e) {
        //Generic exception
        throw new ECSystemException(
            ECTraceIdentifiers.COMPONENT_EXTERN,
            CLASSNAME,
            methodName,
            new Object[] { e.toString() },
            e);
    }

    ECTrace.trace(
        ECTraceIdentifiers.COMPONENT_EXTERN,
        CLASSNAME,
        methodName,
        "CTG properties=" + ctgProp.toString());
}
```

```

/*=====
Get the order information.
=====*/
String orderData = null;
try {
    orderData = getOrderData();
} catch (Exception e) {
    throw new ECSystemException(
        EMessage._ERR_GENERIC,
        CLASSNAME,
        methodName,
        EMessageHelper.generateMsgParms(
            "Error occured when building the orderdata string"));
}

ETrace.trace(
    ETraceIdentifiers.COMPONENT_EXTERN,
    CLASSNAME,
    methodName,
    "Order data=" + orderData);

/*=====
Do the transaction.
=====*/
String transRes = doTransaction(ctgProp, orderData);
ETrace.trace(
    ETraceIdentifiers.COMPONENT_EXTERN,
    CLASSNAME,
    methodName,
    "CICS response information=" + transRes + "");

/* Store the CICS transaction information as a reference
in the log-file, if a rollback would occur. */
EMessageLog.out(
    EMessage._INF_GENERIC,
    CLASSNAME,
    methodName,
    EMessageHelper.generateMsgParms(
        "CICS Transaction information for orderId="
        + getOrderRn()
        + ": CICS Trans response="
        + transRes
        + ""));

/*=====
Validate returned information from CICS.
=====*/
validateResponse(transRes);

```

```

ECTrace.trace(
    ECTraceIdentifiers.COMPONENT_EXTERN,
    CLASSNAME,
    methodName,
    "The response from CICS is valid");

/*=====
   Create transaction response entry in db
   =====*/
try {
    createOrComment(transRes);

    ECTrace.trace(
        ECTraceIdentifiers.COMPONENT_EXTERN,
        CLASSNAME,
        methodName,
        "A row in the ORCOMMENT table has been added "
        +"with the CICS response information");

} catch (Exception e) {
    //The creation of the order comment failed. Output the error message.
    throw new ECSystemException(
        ECMessage._ERR_GENERIC,
        CLASSNAME,
        methodName,
        ECMessageHelper.generateMsgParams(
            "Error occurred when trying to create an order "
            +"comment in ORCOMMENT table for order "
            + getOrderRn()),
        e);
}

ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}

```

Get the CICS Transaction Gateway properties

This method is to retrieve the CICS Transaction Gateway properties that we decided in the design stage to keep stored in the STADDRESS table.

The information is found by using the `findByMemberIdAndNickName` method of the `StoreAddressAccessBean`. At the design stage, we decided to use `CTG-PROP` as a nickname for the store related information, the `URL` variable to be stored in the `Address1` column, the `REGION` variable to be stored in the `Address2` and the `PROGRAM` variable to be stored in the `Address3` column. Retrieving and storing the information will look something like in Example 9-3.

Example 9-3 *getCTGprop* method

```
private java.util.Hashtable getCTGprop()
    throws
        javax.ejb.FinderException,
        java.rmi.RemoteException,
        javax.naming.NamingException,
        javax.ejb.CreateException,
        ESystemException {

    String methodName = "getCTGprop";
    ETrace.entry(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    //Get Store Address entry for NICKNAME='CTG-PROP'
    com.ibm.commerce.common.objects.StoreAddressAccessBean storeAddrAb = null;
    storeAddrAb =
        new com
            .ibm
            .commerce
            .common
            .objects
            .StoreAddressAccessBean()
            .findByMemberIdAndNickName(
                getCommandContext().getStore().getMemberIdInEJBType(),
                "CTG-PROP");

    //Get address1, address2, address3
    java.util.Hashtable stCTGprop = new java.util.Hashtable();

    stCTGprop.put("URL", new String(storeAddrAb.getAddress1()));
    stCTGprop.put("REGION", new String(storeAddrAb.getAddress2()));
    stCTGprop.put("PROG", new String(storeAddrAb.getAddress3()));

    ETrace.exit(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
    return stCTGprop;
}
```

Create the CICS input data

The input data is based on the order reference number and the total price of the order. This method will simply be using the `OrderAccessBean` to get the calculated units, add them together and eventually format the information to be sent to the CICS program into the specified format. For example 12345,312.54.

The order reference number we already have stored as a local variable from the `OrderProcess` command and it is retrieved by using the getter method `getOrderRn`.

The method will look like Example 9-4 when ready. Some trace statements have been added to the example code.

Example 9-4 getOrderData method

```
private String getOrderData()
    throws
        javax.ejb.FinderException,
        java.rmi.RemoteException,
        javax.naming.NamingException,
        javax.ejb.CreateException {

    String methodName = "getOrderData";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    /* Get the order cost. I.e. Totalproduct + Totaltax +
    Totalshipping + TotalAdjustment */
    com.ibm.commerce.order.objects.OrderAccessBean orderAB =
        new com.ibm.commerce.order.objects.OrderAccessBean();

    orderAB.setInitKey_orderId(getOrderRn().toString());

    java.math.BigDecimal orderValue =
        orderAB
            .getTotalProductPriceInEJBType()
            .add(orderAB.getTotalTaxInEJBType())
            .add(orderAB.getTotalShippingChargeInEJBType())
            .add(orderAB.getTotalAdjustmentInEJBType());

    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    //Build and return the order data
    return getOrderRn().toString() + "," + orderValue.doubleValue();
}
```

Do the CICS transaction

The method for connecting and executing the program on the CICS server will be using the J2EE Common Client Interface, as described in the design section. As we are to utilize the ECI adapter, we will use the ECI resource adapter classes in doing so.

These are the steps necessary to do the CICS transaction:

1. Specify the connection properties.
2. Create a connection to the CICS Transaction Gateway.
3. Create an CICS interaction specification that describes the call to be made to the CICS server.

4. Format the input data.
5. Flow the call to the CICS program.
6. Return the result.

We create a method called `doTransaction` that takes the CICS Transaction Gateway properties and the string of information to be passed to the CICS application.

```
private String doTransaction(java.util.Hashtable ctgProp, String orderData)
    throws com.ibm.commerce.exception.ECSystemException {
}
```

Follow the following steps to complete the method:

1. As the code we are writing is to be executed in a non-managed environment, we need to create an `ECIManagedConnectionFactory` object to keep all the information relating connectivity to the CICS Transaction Gateway. Specify the URL for the gateway daemon with the `setConnectionURL` method and the CICS server name with the `setServerName` method.

```
ECIManagedConnectionFactory mcf = new ECIManagedConnectionFactory();
mcf.setConnectionURL((String) ctgProp.get("URL"));
mcf.setServerName((String) ctgProp.get("REGION"));
```

Note: As the CICS Transaction Gateway server listens on the default port, we do not need to define it in the connection properties.

2. Create a `ConnectionFactory` object. Invoking the `createConnectionFactory` method of the `ECIManagedConnectionFactory` object does this. Get the `Connection` object from the `ConnectionFactory` object we just created. The CICS program we will use does not require a user ID and password; if it did we would have to create a `ECIConnectionSpec` object to allow the J2EE component to pass other security credentials.

```
ConnectionFactory cxnf = (ConnectionFactory)
mcf.createConnectionFactory();
```

Note: The J2EE Common Connector Architecture standard implies the use of JNDI for retrieving the `ConnectionFactory` object. For simplicity and to be able to test the application in the WebSphere Test Environment, we create it in the command.

3. Create an `Interaction` object, to the CICS application by invoking the `createInteraction` method from the `Connection` object.

```
Interaction ixn = cxn.createInteraction();
```

4. Create a new `ECIInteractionSpec` object and specify the specifications of the interaction by calling the `setInteractionVerb` method to set to sync and receive mode and use the `setFunctionName` to specify the name of the application to be called. The `ECIInteraction` class defines the specific properties of a call to CICS and is required as a parameter to call the `execute` method.

```
ECIInteractionSpec ixnSpec = new ECIInteractionSpec();
ixnSpec.setInteractionVerb(ixnSpec.SYNC_SEND_RECEIVE);
ixnSpec.setFunctionName((String) ctgProp.get("PROG"));
```

5. Format the input that is to be sent to the CICS program. The input must be a 27 bytes input, thus we do a cast the string to a byte array with a length of 27 with the empty positions to contain the byte value 0. We do not need to update the empty positions in the array as byte is a primitive variable and will be instantiated with the value 0.

```
byte inputCommarea[] = new byte[27];
System.arraycopy(orderData.getBytes(),0,inputCommarea,0,
    orderData.getBytes().length);
```

6. Now we need to create a object that implements the `javax.resource.cci.Record` and the `javax.resource.cci.Streamable` class. The object will contain the data input to the CICS application and is required by the execution method of the transaction. The object class does not yet exist and we need to create it ourselves. We will create a class named `GenericRecord` and how this is done is described well in chapter 5.1 in the *Java Connectors for CICS: Featuring the J2EE Connector Architecture* (SG24-6401-00) redbook and we will implement the same class in our application.

Tip: Refer to the *CICS Transaction Gateway: Gateway Programming* manual for a more detailed description on the input to the CICS application. It is available at

<http://www-3.ibm.com/software/ts/cics/library/manuals/eindex40.htm>

7. Execute the application by calling the `execute` method of the `Interaction` object. Pass the interaction specification and the `GenericRecord` object as parameters. The record can be used to store the response from the CICS transaction, hence it will be used both as a input and output parameter.

```
ixn.execute(ixnSpec, record, record);
```

8. Close the interaction and the connection to the server.

```
ixn.close();
cxn.close();
```

9. Get the response and format the response to a string and return it from the method.

```
String resp = new String(record.getCommarea());
return resp;
```

The method will look something like Example 9-5 when complete with added exception handling and trace statements. Add the following import statements to the class:

```
import com.ibm.connector2.cics.ECIManagedConnectionFactory;
import com.ibm.connector2.cics.ECIInteractionSpec;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.Connection;
import javax.resource.cci.Interaction;
import javax.resource.ResourceException;
import java.io.UnsupportedEncodingException;
```

Example 9-5 doTransaction method

```
private String doTransaction(java.util.Hashtable ctgProp, String orderData)
    throws com.ibm.commerce.exception.ECSystemException {

    String methodName = "doTransaction";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    //Declare the response string
    String resp = null;

    try {
        /* Create and set values for ECI managed connection factory */
        ECIManagedConnectionFactory mcf = new ECIManagedConnectionFactory();
        mcf.setConnectionURL((String) ctgProp.get("URL"));
        mcf.setServerName((String) ctgProp.get("REGION"));

        //Create a connection factory connection object
        ConnectionFactory cxnf =
            (ConnectionFactory) mcf.createConnectionFactory();
        Connection cxn = cxnf.getConnection();

        /* create an interaction with CICS to the program
        specified in the properties */
        Interaction ixn = cxn.createInteraction();
        ECIInteractionSpec ixnSpec = new ECIInteractionSpec();
        ixnSpec.setInteractionVerb(ixnSpec.SYNC_SEND_RECEIVE);
        ixnSpec.setFunctionName((String) ctgProp.get("PROG"));

        /* Create an input commarea from the orderData string
        with the length of 27 bytes.
        */
        byte inputCommarea[] = new byte[27];
        System.arraycopy(
            orderData.getBytes(),
```

```

        0,
        inputCommarea,
        0,
        orderData.getBytes().length);

GenericRecord record = new GenericRecord(inputCommarea);

//Finally execute and flow the request to CICS
ixn.execute(ixnSpec, record, record);

//Close the interaction and the connection
ixn.close();
cxn.close();

//Set the return parameter with the response in a string format
resp = new String(record.getCommarea());

} catch (ResourceException re) {
    throw new ECTSystemException(
        ECTMessage._ERR_GENERIC,
        CLASSNAME,
        methodName,
        ECTMessageHelper.generateMsgParms(
            "A Resource exception occurred when trying to connect to:" +
            ctgProp.toString()),
        re);
} catch (Exception e) {
    throw new ECTSystemException(
        ECTMessage._ERR_GENERIC,
        CLASSNAME,
        methodName,
        ECTMessageHelper.generateMsgParms(
            "Error occurred when executing the CICS transaction"),
        e);
}

//Return the response as the result of the method.
ECTTrace.exit(ECTTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
return resp;
}

```

Validate the CICS response

The validation of the response from the CICS program is a very simple method to implement. The functional requirement states that a CICS response is valid if it contains two colons.

We create a method called `validateResponse` that takes `response`, iterates through it and counts the number of colons. If the number of colons is not equal to 2, we will throw an exception. The method will look something like Example 9-6 when ready.

Example 9-6 validateResponse method

```
private void validateResponse(String transRes)
    throws com.ibm.commerce.exception.ECException {
    String methodName = "validateResponse";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    /* The CICS response is valid if it contains
    the time of the transaction.
    */

    int foundColon = 0;
    for (int i=0;i<transRes.length();i++)
        if (transRes.charAt(i)==':')
            foundColon++;

    if (foundColon != 2)
        throw new ECSystemException(
            ECMessage._ERR_GENERIC,
            CLASSNAME,
            methodName,
            ECMessageHelper.generateMsgParms("CICS transaction did not"+
            " respond with correct information"));

    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}
```

Update the order information

We will create a method called `createOrComment` that will add a row to the `ORCOMMENT` table that will contain the response from the CICS program. We create a new row by creating a new `OrderCommentAccessBean` object with the order reference number and the string to be stored in the table.

The method will look like Example 9-7 when complete.

Example 9-7 createOrComment method

```
private void createOrComment(String transRes)
    throws
        java.rmi.RemoteException,
        javax.ejb.FinderException,
        javax.ejb.CreateException,
        javax.naming.NamingException {
```

```

String methodName = "createOrComment";
ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

//Create an order comment
com.ibm.commerce.tools.optools.order.objects.
OrderCommentAccessBean orderComm =
    new com.ibm.commerce.tools.optools.order.objects.
        OrderCommentAccessBean(
            getOrderRn(), transRes);

orderComm.commitCopyHelper();

ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}

```

9.1.6 Register the command in WebSphere Commerce

In order for the new implementation class to be called instead of the default ExtOrderProcessCmdImpl class, we need to register the implementation class in the CMDREG table.

Update the setting in the database by issuing the following SQL statement:

```

update cmdreg set classname =
'com.ibm.commerce.wc54handbook.commands.DoCicsTransJ2EECmdImpl' where
interfacename='com.ibm.commerce.order.commands.ExtOrderProcessCmd'

```

9.1.7 Deploy the code

In order to make the code executable in a production environment, we will have to verify that the required classes are available in the Application Server environment. As described in 9.1.4, “Pre-requisites” on page 238, the following JAR files needs to be added to the classpath of the enterprise application:

- ▶ cicsj2ee.jar
- ▶ ctgclient.jar
- ▶ ccf2.jar
- ▶ connector.jar
- ▶ screenable.jar

To deploy the code that we have created, refer to Chapter 9 in *WebSphere Commerce Programmer's Handbook Version 5.4*.

9.2 Quick Order

The ToolTech sample store that comes with WebSphere Commerce Business Edition includes a functionality that is commonly implemented in an e-commerce solution. The functionality provides the ability for a user to easily add multiple items to the shopping cart by supplying a list of Stock Keeping Units (SKUs) together with the quantity of the corresponding item in a single request. We refer to this as Quick Order functionality.

With the use of the `OrderItemAdd` command, this functionality is easy to implement in a shop. It takes an indefinite number of SKU numbers and quantities and if they are purchasable to the user the command adds them to the shopping cart and redirects the user a defined URL passed to the command.

Tip: Refer to the WebSphere Commerce documentation on the `OrderItemAdd` command to get a detailed description of the command behavior and dependencies.

Below we provide a brief presentation on how the functionality is implemented in the ToolTech shop together with some design issues.

9.2.1 Quick order flow in the ToolTech store

To get an overview of the logic of the Quick Order functionality, we have analyzed the code in the ToolTech shop and it is summarized in Figure 9-3 together with a brief description below. Refer to the documentation of the `OrderItemAdd` command and view the `QuickOrder.jsp` and the `CatalogItemAdd.jsp` file to get more detailed information on what is actually happening in the ToolTech shop.

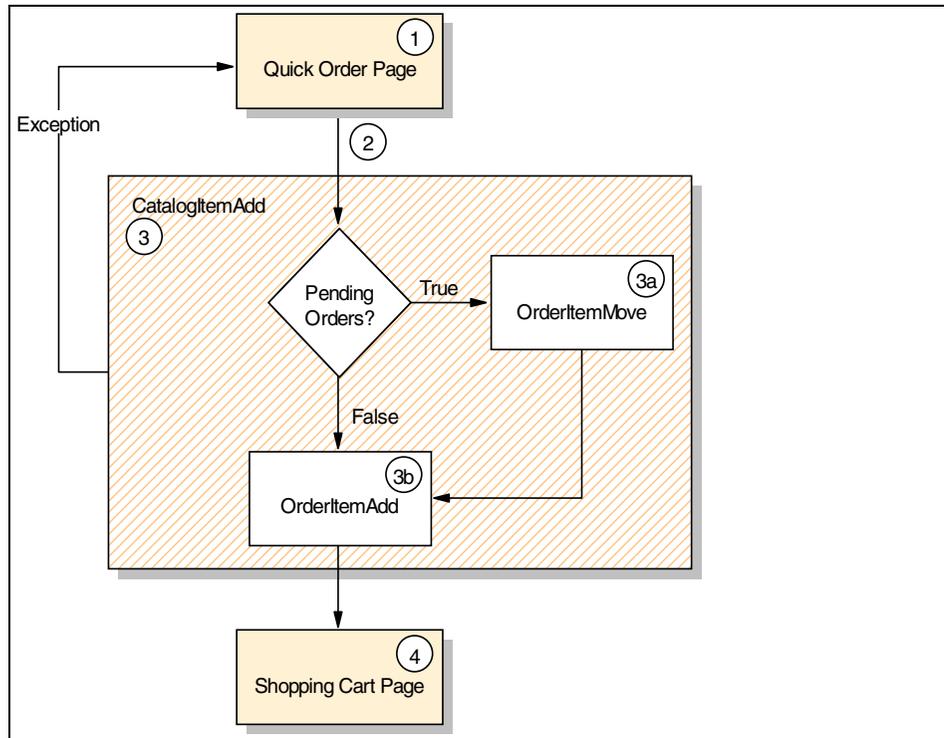


Figure 9-3 The ToolTech flowchart of the quick order functionality

1. The user provides the items SKU numbers and the quantities on the Quick Order page.
2. The user clicks **Order** to process the request. The ToolTech shop is using a customized implementation of the OrderItemAdd command, and instead the information is sent to a view called CatalogItemAdd.
3. The CatalogItemAdd view checks whether the user has more than one pending order associated with his account.
 - a. If true, the CatalogItemAdd view calls the OrderItemMove command with the parameter deleteIfEmpty=*. The OrderItemMove command will move all the pending order items to the current order and delete the empty orders.
 - b. The CatalogItemAdd view calls the OrderItemAdd command and if it is successful it redirects to the provided URL, which is the shopping cart page.

If any exception occurs during the processing within the CatalogItemAdd view, the QuickOrderView is set as an error view and then called.

4. The shopping cart (OrderItemDisplay) page is displayed.

The CatalogItemAdd view further includes control of the errorpage to be shown if an exception occurs. If the CatalogItemAdd view is called from the Quick Order page, the request is redirected back to the Quick Order page and if the view is being called from another page it is redirected to a created view called CatalogItemAddErrorView and with the error message passed along.

9.2.2 Design

When we analyze the code that has been done for the Quick Order functionality in the ToolTech shop, we notice that the main business logic of the functionality is actually in a JSP page. The CatalogItemAdd view requested from the Quick Order page, is a JSP page named CatalogItemAdd.jsp and it contains the main code.

We do not recommend putting business logic in the presentation layer, as has been done with the CatalogItemAdd implementation. It best to have a clean separation between code and presentation, but the CatalogItemAdd.jsp does not actually provide any presentation. The CatalogItemAdd view is a JSP page and JSP pages are normally used for presentation purpose only.

To implement the same functionality, but in a command, create a command wrapper. It would simply be a new command, but it will call other commands from within the performExecute method and set the necessary error views if necessary. Example 9-8 provides an example on how to call a command internally from another command.

Example 9-8 Call a command

```
OrderItemAddCmd oiacmd = (OrderItemAddCmd)
    CommandFactory.createCommand(OrderItemAddCmd.Name,
        getCommandContext().getStoreId());
TypedProperty requestProperties = (TypedProperty)
    getCommandContext().getRequestProperties().clone();

//Modify, if necessary the parameters
requestProperties.put("param1","value1");

oiacmd.setRequestProperties(requestProperties);
oiacmd.setCommandContext(getCommandContext());
oiacmd.execute();
```



Shipping and taxes

This chapter is primarily concerned with shipping and taxes. We will discuss the how to implement shipping by weight in a store. We also explain the check out process as implemented in the ToolTech sample store. We then discuss the display of prices including taxes. The steps to implement this feature will be described and details provided about the tables, commands, and so on that are involved. Discounts will be also described in this chapter. We show how to create a discount for the store and to display the discount in the store

10.1 Shipping by weight

Shipping is how a store handles the delivery of orders to customers. In most cases, products are shipped from a fulfillment center, a separate agency that is responsible for warehousing the store's goods. In order for a WebSphere Commerce store to ship orders, the following must be defined in the database:

- ▶ Shipping mode
- ▶ Shipping arrangement
- ▶ Shipping charge

A shipping charge can be calculated in several possible ways. One of the typical shipping charge is based on weight.

10.1.1 Example of shipping calculations by weight

A department store that has locations in every province, uses shipping by weight. In this example there is only one shipping provider:

The department store assigns products a shipping category according to their weight. Products fit into one of the following weight ranges:

- ▶ 0 to 5lbs
- ▶ 5 to 15lbs
- ▶ 15 to 25lbs
- ▶ 25 to 50lbs
- ▶ 50 to 100lbs
- ▶ More than 100 lbs

The shipping charge is based on the weight of the product. No other shipping charges apply. Table 10-1 shows an example of weight based shipping charges.

Table 10-1 Shipping charges based on weight

Jurisdiction	0 to 5 lbs	5 to 15 lbs	15 to 25 lbs	25 to 50 lbs	50 to 100 lbs	More than 100 lbs	Cost per product	Cost per item
United States	4.00	4.00	8.00	8.00	15.00	25.00	0.00	0.00

In this example when a shopper buys a microwave oven, the shipping charge is \$8.00 calculated as \$5.00 for 15 to 25lbs shipping times 1 microwave oven.

10.1.2 Shipping database assets

ToolTech is the business-to-business (B2B) online hardware store provided with WebSphere Commerce. One of the features included in the ToolTech sample store is weight-based shipping.

The ToolTech store provides all the pages and features necessary for a functioning B2B online store. ToolTech is packaged with WebSphere Commerce as a store archive. To view the sample store create a new store archive based on ToolTech using the Store Services tools, then publish the archive to the WebSphere Commerce Server.

For more information about publishing a store archive refer to the WebSphere Commerce Version 5.4 Online Help.

The ToolTech shipping database assets are stored in the following XML files:

- ▶ shipping.xml
- ▶ store-catalog-shipping.xml
- ▶ store-defaults.xml
- ▶ shipfulfill.xml

The ToolTech shipping assets are divided into:

- ▶ Jurisdictions
- ▶ Shipping modes
- ▶ Calculation codes
- ▶ Calculation rules
- ▶ Calculation scale
- ▶ Calculation range
- ▶ Calculation lookup
- ▶ Calculation combinations
- ▶ Shipping fulfillment

Jurisdictions

The shipping.xml file identifies jurisdictions for shipping.

Jurisdiction is defined in the JURST table. The Table JURSTGROUP assigns the jurisdiction to a group and subclass. Jurisdiction and the group are assigned to the same group in the table JUSTPREL.

Shipping modes

A shipping mode is a combination of a shipping carrier and its shipping service.

Calculation codes

The calculation codes are necessary for the calculation of shipping charges, sales tax, weight ranges, shipping tax and discounts. The shipping.xml contains the calculation for shipping.

Field displaylevel contains the amount calculated and the source. Values are:

- ▶ 0 = Order Item
- ▶ 1 = Order
- ▶ 2 = Product
- ▶ 3 = Item
- ▶ 4 = Contract

Calculation rules

The CALRULE table stores the calculation rules for shipping, and shipping by weight. The flag field contains the CalculationCodeQualifyMethod value of the specific CalculationCode.

- ▶ 0 = the method will not be invoked
- ▶ 1 = the method will be invoked

Calculation scale

A calculation scale is the set of ranges that applies to the calculation. For example shipping cost with a set of weight ranges:

- ▶ 0 and 5 kg costs \$10.00
- ▶ 5 and 10 kg costs \$15.00

The CALSCALE table stores the scale code for shipping and shipping by weight, one per order and one per item.

Calculation range

The range for the scale codes is stored in the CALRANGE table:

- ▶ calmethod_id_10 = shipping per order
- ▶ calmethod_id_11 = shipping per item

Calculation lookup

Calculation lookup values are the values associated with the calculation scale. For example, the calculation lookup values for a product with a set of weight ranges from:

- ▶ 0 to 5 kg costs \$10.00
- ▶ 5 to 10 kg costs \$15.00

would be \$10 to \$15 . There is also one lookup value per currency for a given CALRANGE ID. The CARLOOKUP table defines the lookup ID and value.

Calculation combinations

Calculation rules and the scale ranges are combined in the CRULESCALE and calculation methods and rules are combined in the STENCALUSG table.

The field usageflag controls how the OrderPrepare command uses the calculation:

- ▶ 1 = use - use this CalculationUsage.
- ▶ 2 = check - throw an ECAApplicationException if this calculation does not produce a value for an order item.

Shipping fulfillment

Shipping fulfillment assets associate a shipping jurisdiction group to the calculation rules, and a fulfillment center to the ship mode, for the store. The shipping fulfillment information is stored in the SHPJCRULE and SHPARRANGE tables.

For more information about shipping database assets of the sample store refer to the product documentation for ToolTech or to the WebSphere Commerce Version 5.4 Online Help, or to the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*.

10.1.3 Adding shipping by weight charges

The sample stores, InFashion, WebFashion, WebAuction, and NewFashion do not include shipping by weight charges. If you are creating your store using one of these samples, you will have to edit some of the store database asset XML files.

Changing shipping settings

Make the necessary changes of the shipping settings using the Shipping notebook in Store Services. Follow this sequence:

1. Change shipping providers
2. Change provider display name
3. Change shipping zones
4. Change shipping categories
5. Change shipping rates
6. Change weight ranges
7. Change weight rates

After changing the shipping settings the values will be updated the next time you publish the store.

For more information about changing the shipping settings, refer to the IBM WebSphere Commerce Version 5.4 Online Help.

Changing store database assets

The Shipping notebook does not edit all database fields related to shipping. For more information, see “Changing store database assets” in the WebSphere Commerce Version 5.4 Online Help.

To include shipping by weight charges in one of the sample stores you have to localize the store archive file for your store (for example: mystore.sar) . The directory where SAR files are located is:

```
<WCS_DIR>\CommerceServer\instances\instancename\sar
```

1. Open the store archive file using a ZIP program.
2. Search for the store-catalog-shipping.xml and open the file
3. Add the ship by weight calculation code (0) to the file as shown in Example 10-1:

Example 10-1 Shipping by weight calculation code

```
<catencalcd  
  calcode_id="@calcode_id_0"  
  catencalcd_id="@catencalcd_id_0"  
  store_id="@storeent_id_1"  
>
```

For more information about the CATENTCALCCD table refer to the IBM WebSphere Commerce Version 5.4 Online Help.

4. Add the information shown in Example 10-2 for each catentry (products, items, bundles and packages) in the catalog:

Example 10-2 Shipping by weight

```
<catentship  
  catentry_id="the product or item id"  
  noninalquantity="1.0"  
  quantitymultiple="1.0"  
  weight="the weight of the product or item"  
  weightmeasure="LBR"  
  quantitymeasure="C62"  
>
```

For more information about the CATENTSHIP table refer to the IBM WebSphere Commerce Version 5.4 Online Help.

5. Save the file and ensure the updated file is in the store archive file.
6. If you want to add information after publishing the store, you can use the Product Management Tools. These will only update the data in the database and not alter the database assets in the SAR file. Details about how to use the Product Management tools are described in the WebSphere Commerce Version 5.4 Online Help.

For more information about creating and changing shipping assets refer to the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*, Chapter 18.

Loading data

There is a simple way of loading data into WebSphere Commerce Server database using the Loader package commands:

To load data into your WebSphere Commerce Server database, run the Load command:

1. Create a working directory. For example:
`C:\template\data`
2. Make sure that the XML file is in a location is included in the same path:
That is place store-catalog-shipping.xml in C:\template\data\.
3. From a Window command prompt, enter the following command:
`cd <WCS_DIR>\CommerceServer\bin`
4. Be sure to back up the WebSphere Commerce Server database.
5. From the Windows command prompt, enter the following command:
`massload -dbname mall -dbuser databaseuser -dbpwd databasepw -infile "C:\template\data\store-catalog-shipping.xml" -method sqlimport`
6. Check the database and the tables CATENTCALCD and CATENTSHIP

More information about loading data into the WebSphere Commerce Server database refer to the *IBM WebSphere Commerce Store Developer's Guide Version 5.4* and to the WebSphere Commerce Version 5.4 Online Help.

10.1.4 Use case for order check out

This section describes the use case and interactions when a shopper and order decides to check out and order from the store. The use case is initialized by clicking the **Check Out** button. The sequence that occurs as a result is:

1. A bill address page is displayed. The customer has the choice to select one of the addresses in their address book for the billing address or to create a new address. If the customer clicks **Create new address**, a new address can be added.
2. The system sets up the selected address as the billing address for the order.
3. The system displays the shipping address page. The customer can select an existing address or to create a new address. The system sets up the selected address as the shipping address of the order.
4. The customer select a shipping method. This process is handled in the shipping.jsp.

The shipping JSP uses the following commands:

- AddShipModeView
- OrderItemDisplay

The shipping JSP uses the following beans:

- OrderBean
- OrderItemAccessBean
- ShipModeAccessBean

By clicking **Check Out** in the shopping cart page the customer moves through a series of checkout pages. The OrderItemDisplay command decides which page will be loaded. The command OrderItemDisplay returns the OrderItemDisplay.jsp.

This OrderItemDisplay.jsp includes different JSP files based on the page parameters. If a value for the page is shipmethod, the third page loaded in the check out sequence will be the shipping method page (shipping.jsp).

The shipping method page includes a form where the customers can select the shipping method. The action for the form is set to AddShipModeView. This command is registered in the VIEWREG table to associate with AddShipMode.jsp

5. The form with AddShipModeView is submitted and the AddShipMode.jsp is displayed.

The AddShipMode.jsp uses the following commands:

- OrderItemUpdate
- OrderPrepare

6. The shipping method is updated for the order item by the OrderItemUpdate command.
7. The OrderPrepare command is called to preprocess the order. The OrderSummary is displayed if the status parameter is set to P. In this case the OrderDisplayPending.jsp is displayed.

The shipping.jsp displays the cost structure and approximate delivery time for each shipping method. The information are stored in the SHPMODEDSC table.

If you want to modify the shipping charges in the database, you have to update the description in the SHPMODEDSC.

Commands

OrderItemDisplay

The OrderItemDisplay command lists all order items which are in pending status. The parameters of the command are shown in Figure 10-1.

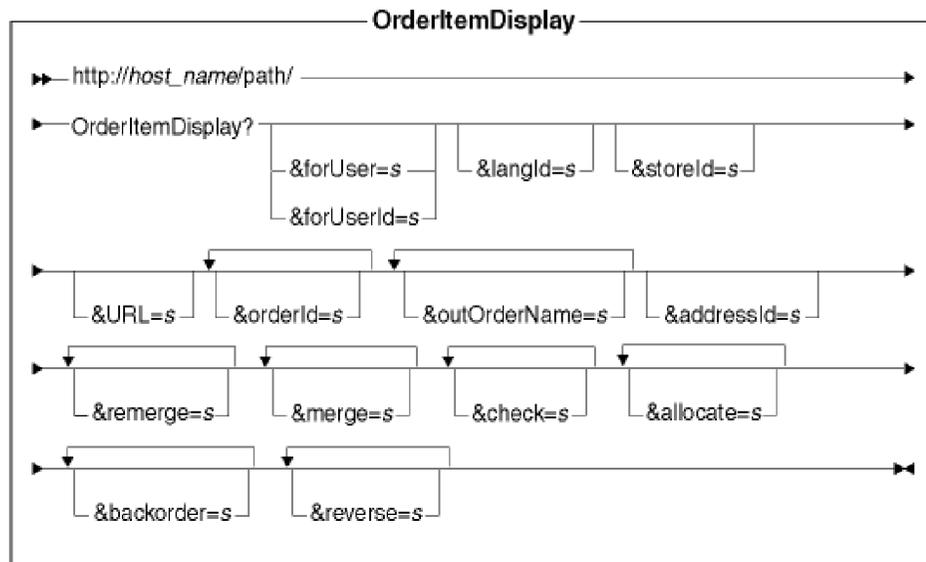


Figure 10-1 OrderItemDisplay

OrderItemUpdate

This command can add products and items into one or more orders into the order list. It can also update OrderItems in an existing order. The parameters of this command are shown in Figure 10-2.

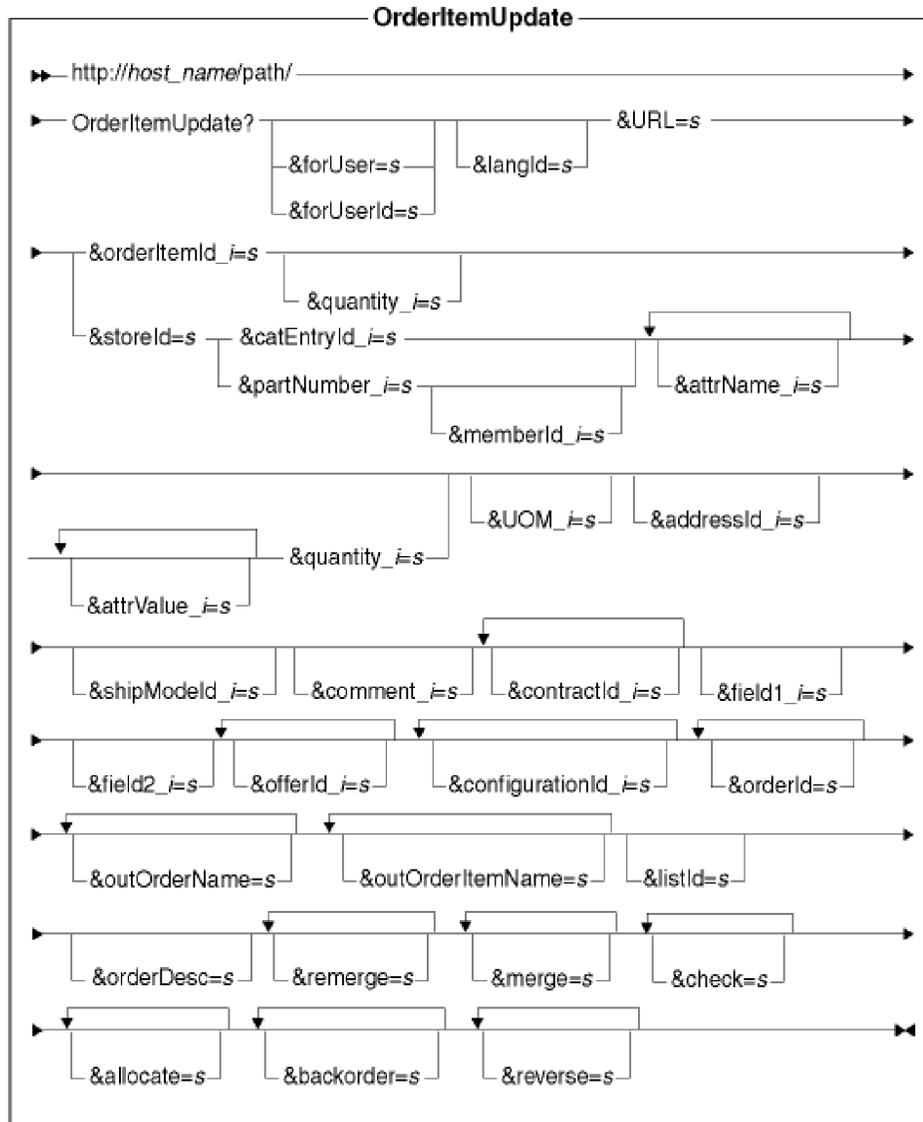


Figure 10-2 OrderItemUpdate

OrderPrepare

The command prepares an order by determining shipping charges for an order. If an order reference number is not specified, all current pending orders will be prepared for the current customer at the given store. The parameters of this command are shown in Figure 10-3.

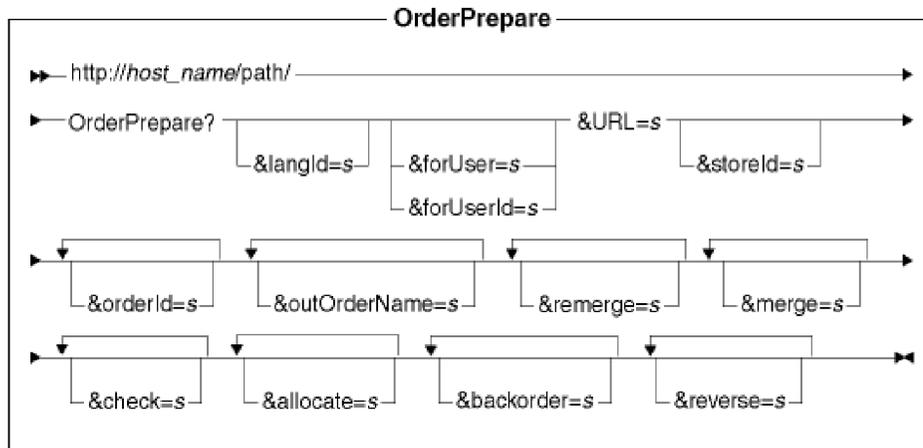


Figure 10-3 OrderPrepare

For more information about commands, beans and tables refer to the WebSphere Commerce Version 5.4 Online Help.

10.2 Prices including taxes

Each type of tax used in your store should have its own category. Default tax categories are set up for you. Sales tax is charged on the total amount of the order. Shipping tax is charged on the shipping charges for the total order. If you want to include the tax amount in the price that displays for the product you have to make same changes to the store samples. If you do not want include tax amount in price, the tax amount will display separately.

This chapter includes information about tax types, examples of tax calculations, details about tax assets in one sample store, and information on changing tax settings. We describe how to display tax as part of the display price in the ProductDisplay.jsp.

10.2.1 Definition of tax types

A store typically collects two type of taxes: sales or use tax, and shipping tax. Each tax category has one tax type. Although each tax category may only be of one tax type, several different tax categories may belong to the same tax type.

The following tax calculations are based on sample scenarios. All examples are hypothetical.

For this example imagine a department store located in Ontario, Canada, and that the store has customers from other jurisdictions, where taxation laws may vary. Table 10-2 outlines sample jurisdictions, the store's tax categories, and the tax rates in percentages:

Table 10-2 tax example

Jurisdiction	default	shipping	Federal tax	State tax
USA, California	0.00	0.00	6.00	8.00
USA, other	0.00	0.00	6.00	7.00
Canada , Ontario	0.00	0.00	7.00	8.00
Canada, other	0.00	0.00	7.00	7.00

Table 10-3 illustrates the tax codes:

Table 10-3 tax codes

tax code	default	shipping	federal	state
clothings	yes	no	yes	yes

A shopper from Ontario, Canada orders two pairs of women's jeans. Jeans fall under the tax code clothing, which is composed of the default, federal and state taxes. In the jurisdiction CANADA, Ontario, the values for these taxes are 0.00, except for federal tax, which is 7 percent, and state tax, which is 8 percent. As a result the total tax charge is \$21.00: ($\70.00 per pair of jeans * 7% federal tax * 2 pairs of jeans) + ($\$70.00$ per pair of jeans * 8% state tax * 2 pair of jeans) = \$21.00.

10.2.2 Tax database assets

WebSphere Commerce samples stores keeps tax database assets in the following XML files:

- ▶ taxfulfill.xml
- ▶ store-catalog-tax.xml
- ▶ store-defaults.xml
- ▶ tax.xml

The tax database assets are divided into the following sections:

- ▶ Jurisdictions
- ▶ Tax categories
- ▶ Calculation codes
- ▶ Calculation rules
- ▶ Calculation scale
- ▶ Calculation range
- ▶ Calculation lookup
- ▶ Calculation combinations
- ▶ Tax fulfillment

Jurisdiction and groups

Jurisdictions are geographical regions or zones representing a country or region, province or territory, or zip code range, to which you sell goods. Jurisdictions are grouped together to form jurisdiction groups.

WebSphere Commerce has two types of jurisdiction: shipping and tax jurisdiction. The tax jurisdiction is divided into groups.

The tax.xml file identifies jurisdictions. Jurisdictions are defined in the JURST table, JURSTGROUP assigns the jurisdiction to a group and subclass, and JURSTPREL assigns the jurisdiction and jurisdiction group to the same subclass.

Tax categories

Each type of tax used in the sample store has its own tax category such as state or region, provincial, or federal. Each tax is classified as a sales tax or shipping tax. The TAXCGRY table stores the tax categories as shown in Figure 10-4.

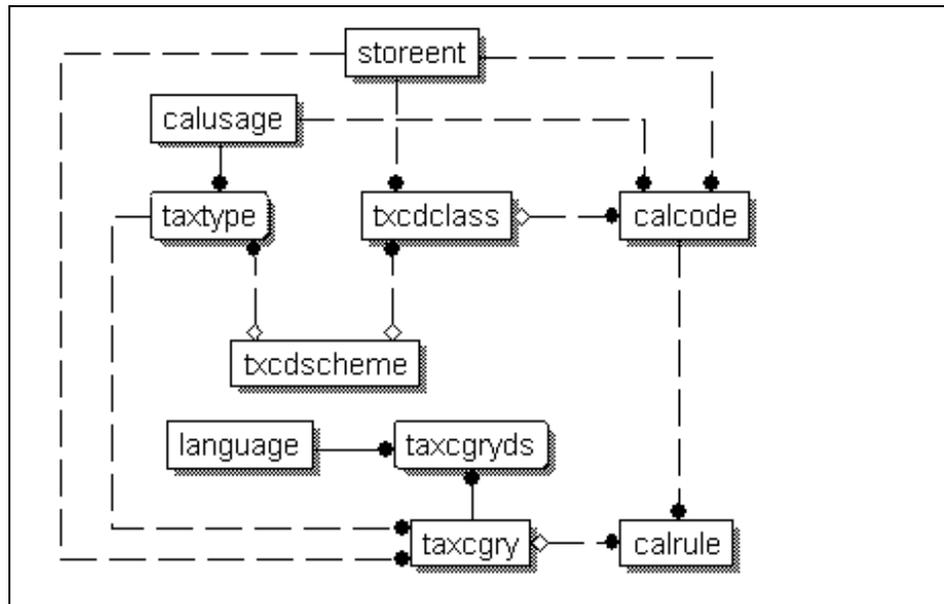


Figure 10-4 Tax type and category data model

Calculation codes

Calculation codes are used to calculate tax charges - this is how tax is calculated for an order item. Tax calculated for an order item must use either a sales tax or a shipping tax calculation code.

Only one tax calculation code of each tax type is supported for a catalog entry or a group of catalog entries.

Sales tax is based on the net price and shipping tax is based on the shipping charge.

The tax.xml file contains all the calculation codes for taxes. The CALCODE table stores the calculation codes for taxes. The displaylevel field has a number displaying what amount was calculated. The numbers that can be displayed are shown below:

- 0 = Order Item
- 1 = Order
- 2 = Product
- 3 = Item
- 4 = Contract

Calculation rules

One calculation code has a set of calculation rules which define how the calculation will be done. The CALRULE table stores the calculation rules for the tax category. The flag field specifies whether the CalculationCodeQualifyMethod value of the specific CalculationCode should be invoked. A tax calculation rule is associated with a tax category, a jurisdiction group, and a fulfillment center. This defines conditions for how the calculation rule is used. These relationships are shown in Figure 10-5.

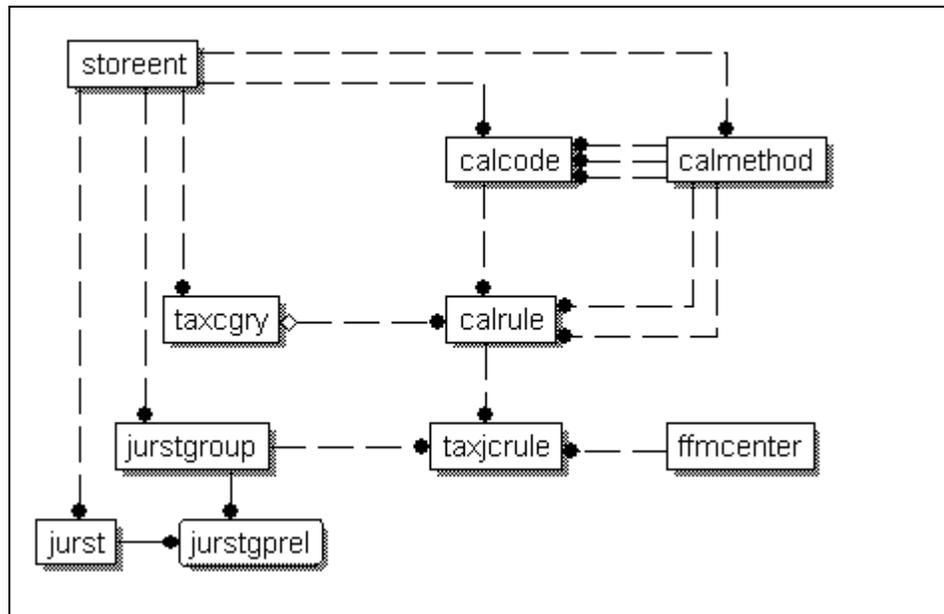


Figure 10-5 Calculation rule - tax data model

Tax fulfillment

Tax fulfillment assets associate a tax jurisdiction group to a fulfillment center and a calculation rule to both. The tax fulfillment information is stored in the TAXJCRULE table and can be seen in the taxfulfill.xml file:

The taxjcrule_id field generates a unique key for each association of jurisdictions, fulfillment centers, and calculation rules.

For more informations about calculation scale, calculation range, calculation lookup and calculation combinations refer to the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*, chapter 19 or to the WebSphere Commerce Version 5.4 Online Help.

10.2.3 Implementation prices including taxes

InFashion sample store

InFashion is one of the business-to-consumer online fashion stores provided with WebSphere Commerce. The sample store is packaged with WebSphere Commerce as a store archive, and as a result, no further installation is necessary. All that is required to view the sample store is to create a new store archive based on InFashion using the Store Services tools, then publish it to the WebSphere Commerce Server.

In this section we describe the changes necessary to the InFashion sample store in order display tax as part of the display price.

For more information about setting up the InFashion sample, see "Creating a store archive using Store Services" in the WebSphere Commerce Version 5.4 Online Help.

Changing tax settings

To change the necessary tax settings you can use the tax notebook in Store Services:

1. On the machine where WebSphere Commerce is installed, select the **Start -> Programs -> IBM WebSphere Commerce -> Store Services**.

On the machine where WebSphere Commerce is installed, or on a client machine on the same network as the WebSphere Commerce machine you can also access Store Services using the following Web address in your browser:

`https://<hostname>:8000/storeservices`

2. The Logon page is displayed in the browser. You must log in to the Store Services with administration rights.

3. In the **User name** field, type your user name.
4. In the **Password** field, type your password.
5. From the **Display language for your store field**, select the preferred language in which to work. See Figure 10-6.

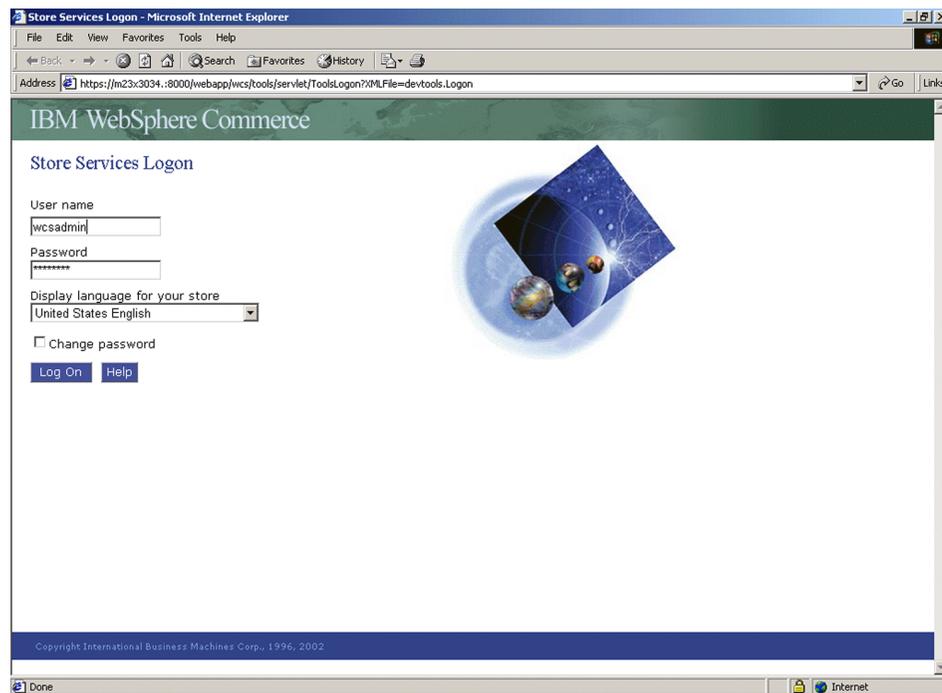


Figure 10-6 Log in to Store Services

6. Click **Log On**.
7. The Store Archive list displays.
8. From the Store archive list select your sample Store (for example Mystore.sar) and click **Tax**.

Changing tax jurisdiction

The sample store archives define the World tax jurisdiction, which cannot be deleted using the tax notebook. If your specified country or region and state or province combination do not match, the World jurisdiction is used instead. For our example we still used the default values.

Changing sales taxes

Default sales tax codes are set up for you. You can use this default sales tax code, or you can delete it and create a new one.

The sample stores and the tax notebook only support a single sales tax code. To add more tax codes to your stores, see details in the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*.

To change the Sales taxes use the following steps:

1. From the left navigation frame of the tax notebook, click **Sales Tax**. To add a code, do the following:

In the New sales tax code field, type the name of a sales tax code for example: Tax Code 2

2. Click **Add**. The tax category displays in the Defined sales tax codes list
3. To set the default tax code, from the Define sales tax codes list, select the code **Tax Code 2**, then click **Set as Default**. The default sales tax code is applied to all products that are not currently assigned to a sales tax code. To delete a code, in the Defined sales tax codes list, select the code, then click **Remove**. If you wish to complete another task in the tax notebook, click the appropriate page in the left navigation frame. Figure 10-7 shows the tax notebook.

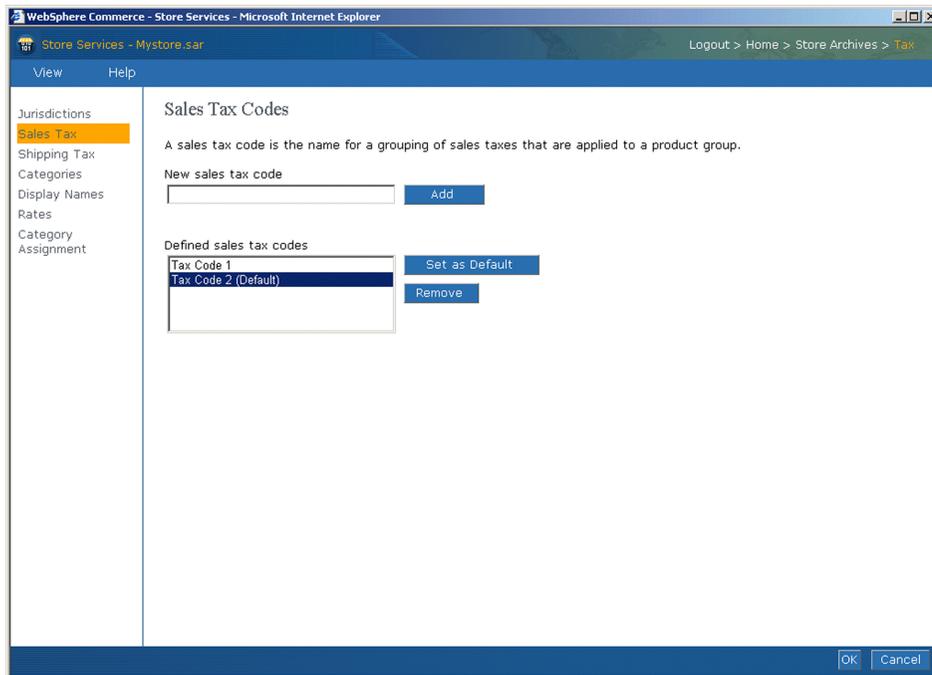


Figure 10-7 Changing sales tax code

Changing tax categories

The next step to create a new tax category. To do this:

1. Click **Categories** in the left navigation frame of the tax notebook.
2. In the New tax category field, type the name of the tax category. For example enter:

VAT

Be careful when choosing the name of your tax category. You can remove the category if it has not been assigned to any products, but you cannot change the name of the tax category.

3. Select the tax type of the category. Each category must be either a sales tax or a shipping tax type. Sales tax is charged on the total amount of the order. Shipping tax is charged on the shipping charges for the total order.

Select **Sales Tax**.

4. Select **Include tax as display price**.

The tax amount will be included in the product display price. If this option is not selected, the tax amount will display separately.

Attention: To display tax as a display price on the JSP file, you have to make some changes. The steps to change the ProductDisplay.jsp example are described in this chapter.

5. Click **Add**. The tax category displays in the Defined tax categories list. See Figure 10-8.

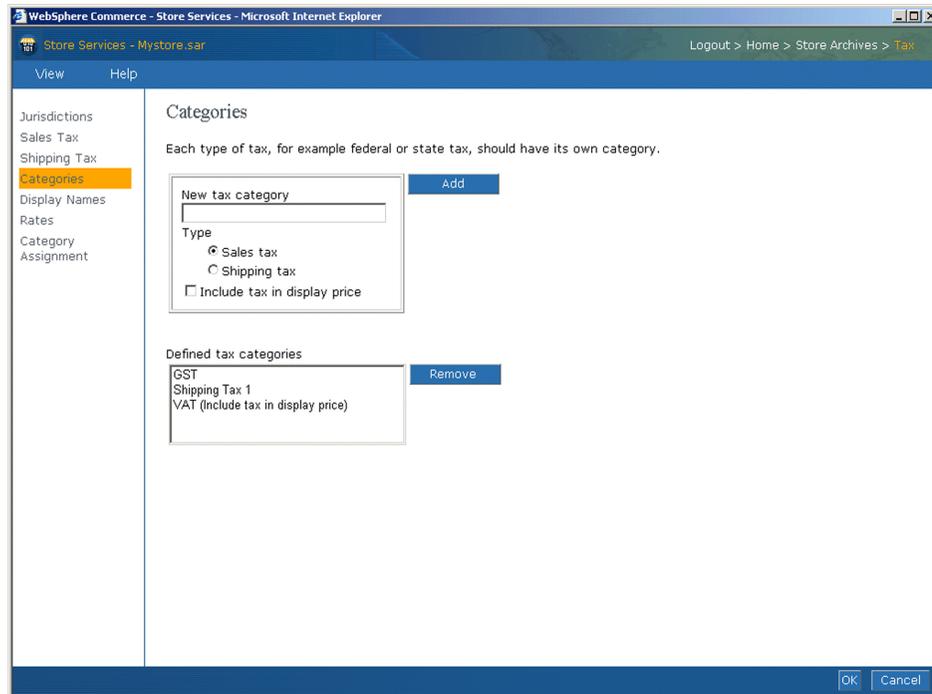


Figure 10-8 Changing tax category

Changing display names

After you define your tax category, you must define the name for each category that will display to your customers. Refer to the WebSphere Commerce Version 5.4 Online Help for details on how to change the display names.

Changing rates

1. From the left navigation frame, click **Rates**.
2. The tax rates table contains a default tax, which has an initial value of 0.00. In the column VAT type a tax rate. For example:

7.0000

Leave the value as zero for the shipping tax. The tax rate averages 7% of the display price of the product.

3. Click **OK** to save your settings. See Figure 10-9

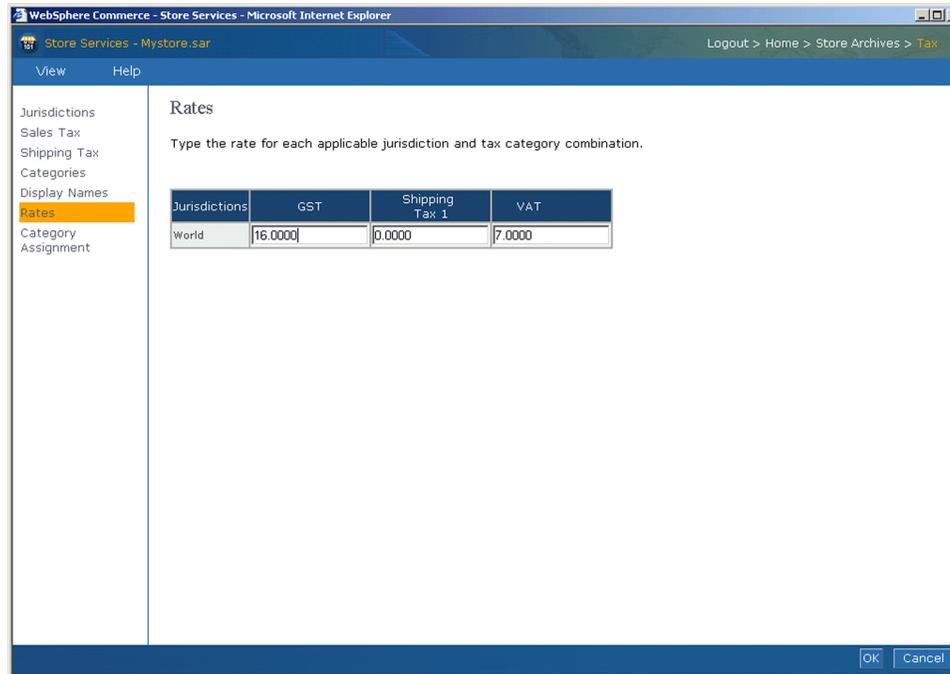


Figure 10-9 Changing tax rates

Changing category assignments

1. From the left navigation frame, click **Category Assignment**.
It also contains a default tax, GST and a default shipping tax.
2. Assign tax category VAT to the appropriate tax codes Tax Code 2, by selecting the corresponding checkbox in the column VA. This is shown in Figure 10-10

Tax Codes	GST	Shipping Tax 1	VAT
Tax Code 1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Shipping Tax Code 1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Tax Code 2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 10-10 Changing category assignments

Publish store database assets

After the changing of the tax settings you have to publish only the store database assets. The steps are:

1. Back up your WebSphere Commerce database
2. Before republishing a store, delete the files from following directory:
`<WCS_DIR>\CommerceServer\instances\instancename\cache\storeid\`
In order to avoid problems, delete only the content under the storeid folder. In the store development phase, you should disable caching. To do this, open the Caching panel of the Configuration Manager, and ensure that **Enable Cache** is deselected.
3. From the Store Archive list, select the store archive you wish to publish (for example Mystore.sar)
4. Click **Publish**. The Publish Store Archive page displays.
5. Select only the check boxes for the **Publish store database assets** to update the WebSphere Commerce database.
6. Click **OK** to publish the store database assets to the WebSphere Commerce Server.

If you have problems to publishing the store database assets refer to the WebSphere Commerce Version 5.4 Online Help, see Chapter “Troubleshooting Publishing”.

Changing ProductDisplay.jsp

After publishing the store you have to change the JSP file where you want to display prices including tax. We describe the changes to the ProductDisplay.jsp from the sample store. The steps are:

1. Change to the following directory:
`<drive>\WebSphere\AppServer\installedApps\WC_Enterprise_App_instance_name.ear\wcstores.war\WEB-INF\classes\storedir`
In our example the name of the instance Mystore.
2. Backup the ProductDisplay.jsp
3. Open the ProductDisplay.jsp and include the code shown in Example 10-3:

Example 10-3 Import packages in example JSP

```
<%@ page import="com.ibm.commerce.price.beans.*" %>
<%@ page import="java.math.*" %>
```

To include the tax in the display price of the product you have to get the amount of the CalculationPrice and also the amount of the tax. The tax amount must be added to the CalculationPrice and be formatted again. See Example 10-4

Example 10-4 Get the amount of the calculationprice and the tax

```
<%  
// Display tax as part of the display price  
  
BigDecimal totalPriceWithTax =  
product.getCalculatedContractPrice().getAmount().  
add(product.getDisplayTaxes().getCategoryAmount());  
FormattedMonetaryAmountDataBean formattedAmount = new  
FormattedMonetaryAmountDataBean();  
formattedAmount.setAmount(totalPriceWithTax);  
com.ibm.commerce.beans.DataBeanManager.activate(formattedAmount,request);  
%>
```

4. To display the price including taxes include the code shown in Example 10-5

Example 10-5 Total price with tax

```
<font class="price">Total Price with Tax :  
<%=formattedAmount%></font><br>
```

5. To display the price without tax include the code shown in Example 10-6

Example 10-6 Price without tax

```
<font class="price">regularPrice  
:<%=product.getCalculatedContractPrice()%></font><br>
```

6. Save the ProductDisplay.jsp and close the file.

7. You can launch your store by entering the following:

```
https://host_name/webapp/wcs/stores/store_directory/index.jsp
```

Or you can enter the following URL to display the ProductDisplay.jsp:

```
https://local_name/webapp/wcs/stores/servlet/ProductDisplay?catalogId=catalog_id&storeId=store_id&productId=product_id&langId=-1
```

You will get the display shown in Figure 10-11 with both prices appearing.

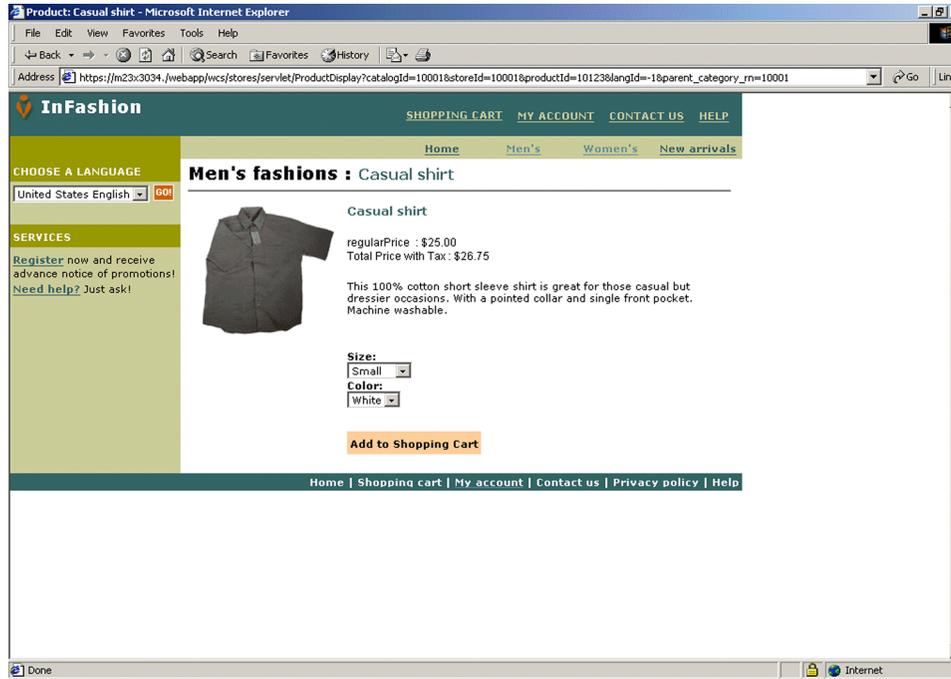


Figure 10-11 Prices with and without tax

8. If the ProductDisplay.jsp fails for any reason check the log files in the following path:
`<WCS_DIR>:\WebSphere\CommerceServer\instances\instance_name\logs\`
 The ecmsg.log file produces system files with names like
`ecmsg_instance_name_2002.05.01_18.17.47.768.log`
9. After checking to logs, correct any errors, and try to go to the ProductDisplay page of you store again.

10.3 Discounts

Discounts allow you to offer customers price incentives to promote a purchase. You are allowed to offer percentage discounts or fixed-amount discounts. They can apply to specific products or to the total purchase.

Discounts can be either active, or inactive. They are set as active by default when created, but can be deactivated at any time using the WebSphere Commerce Accelerator. If you change a discount from active to inactive, you must propagate the discount to the production server for the change to take effect.

The WebSphere Commerce Accelerator include a wizard to create discounts. Once created, the discounts must be deployed to the production server. It is also possible to create discounts with the Loader utility.

10.3.1 Discount types

In WebSphere Commerce they are different types of discounts. Using either a simple discount or a multirange discount, you have four discount types:

- ▶ Percentage off total purchase
- ▶ Amount off total purchase
- ▶ Percentage per product
- ▶ Amount off per product

For more information about discount types refer to the section on discount types in the WebSphere Commerce Version 5.4 Online Help.

10.3.2 Discount assets

The primary method of creating discounts is using the discount wizard in WebSphere Commerce Accelerator.

Discounts can also be created by using XML files and then loaded by the Loader package or published by Store Services. For more information on to create discount assets refer to WebSphere Commerce Version 5.4 Online Help or to *IBM WebSphere Commerce Store Developer's Guide Version 5.4*.

Discount calculation code

Figure 10-12 shows the discount structure in the order model implemented by WebSphere Commerce Server.

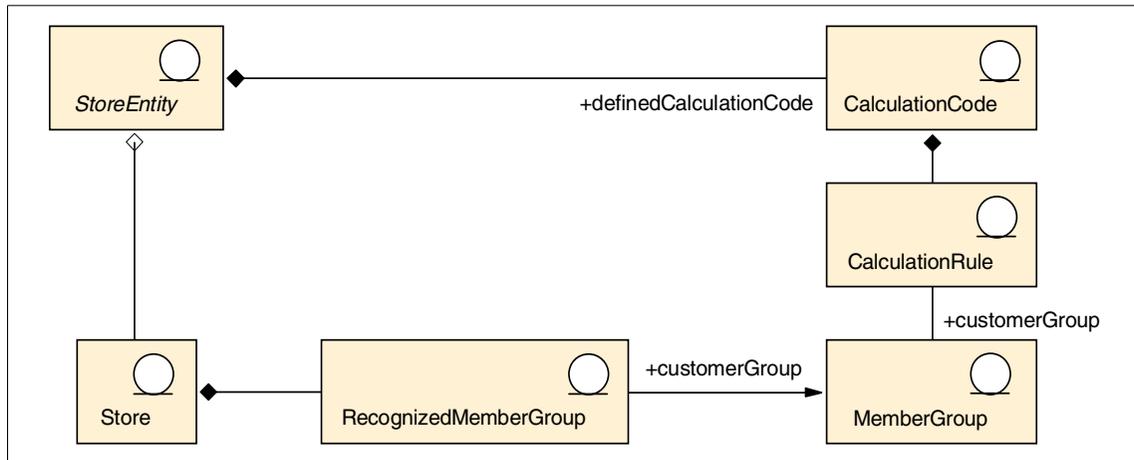


Figure 10-12 Order object model

A discount calculation code indicates how the discount is calculated for order items. Each row in the table CALCODE represents a calculation code. A calculation code belongs to a store entity. More than one calculation code can be defined in one store entity.

Discounts can be offered all the time or only for a defined time period. The discount calculation code can also be associated with one or more member groups and this is defined in the CALCODEMGP table. The usage of a calculation code by members in a certain member group, is defined in the CALRULEMGP table. It can also be attached to one or more catalog entries or to a catalog group.

Catalog entries or catalog groups can have more than one discount calculation code. If there is more than one calculation code for an order, the discount calculations are performed in ascending sequence of their calculation code sequence attributes.

Order items are grouped for calculation in one of four ways:

- ▶ Per trading agreement
- ▶ Per Product
- ▶ Per offer
- ▶ Per shipping address

For more information, see the WebSphere Commerce Version 5.4 Online Help section on “Discount and discount codes”.

10.3.3 Creating a discount in a sample store

In this section we show how to create a simple discount for a new customer profile for registered users. The discount type is “Amount off per product”.

Creating a new customer profile

A customer profile incorporates registration information, demographics, address information, customer culture, purchase history, and other miscellaneous attributes which define a dynamic group of customers or accounts. Customer profiles serve as targets for discounts.

Customer profiles are created and edited using the customer profile notebook in the WebSphere Commerce Accelerator. To create a customer profile:

1. Log into WebSphere Commerce Accelerator.
2. Choose **Marketing -> Customer Profiles**.
3. Click **New**. The customer profile notebook is displayed.
4. Type a name into the customer profile text field and also provide a description.
For example:
IBMWCS54 - Testgroup
5. Set the registration status to **Registered**.
6. Click **OK** to save the profile. The customer profile is displayed in the customer profile page, as shown in Figure 10-13.

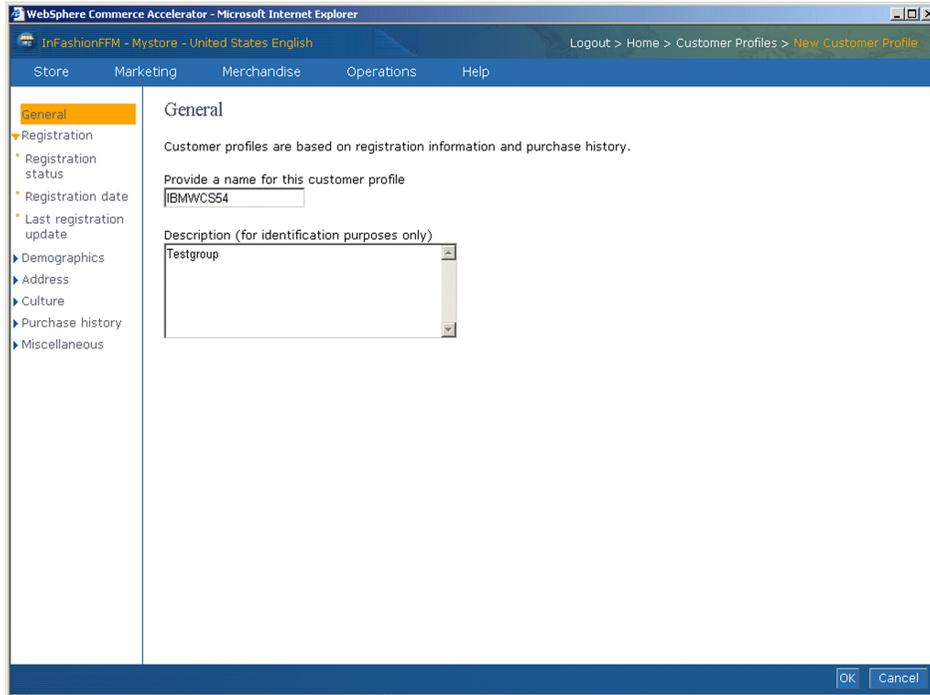


Figure 10-13 Creating a new customer profile

After you have created the customer profile, the new profile should appear in the list of the customer profiles as shown in Figure 10-14.

Name	Description	Last Update	Modified By
<input checked="" type="checkbox"/> IBMWCS54	Testgroup	May 2, 2002 2:57 PM	wcsadmin

Figure 10-14 Sample customer profile

All customers which fit in the profile can also be shown by selecting the profile and clicking **Customers**.

Creating a new discount

In the first step you have created a new customer profile, now we create a new discount for this group by taking the following steps:

1. Choose **Merchandise -> Discounts**.
2. Click **New**.

3. Fill in the name and the description in the discount general Information page.
For example:
Name : IBMWCS54Discount
Description: Discount of IBM
4. Select **This discount is always in effect** and click **Next**.
5. In the customer groups windows select **Assign this discount to specific customer profiles**.
6. Now you are able to select the new customer profile IBMWCS54 by clicking **Add** as shown in the Figure 10-15.
7. Click **Next**.

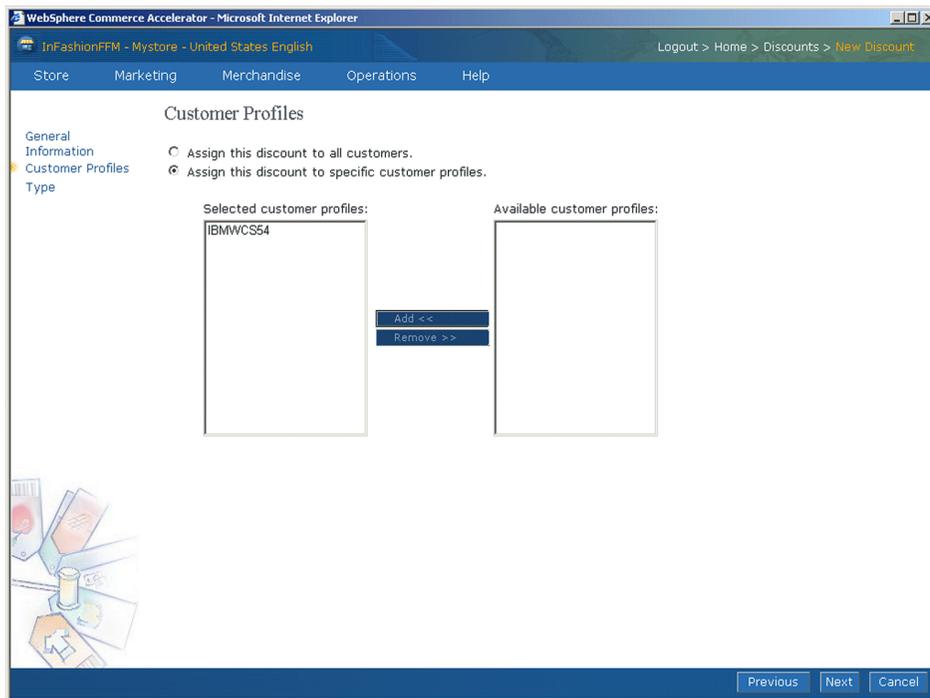


Figure 10-15 Customer groups

8. The discount type window appears. Select **Amount off per product** and click **Next**.
9. On the simple display page enter the amount for you discount, for example:
5 USD.
10. Select **Specify the minimum number of products necessary to qualify for this discount**.

- Fill in for the number of products that a customer must purchase to get the discount. We selected **All qualifying product must be the same** to get the discount only on the same products. Click **Finish**. You will get a message that the discount has been successfully created.

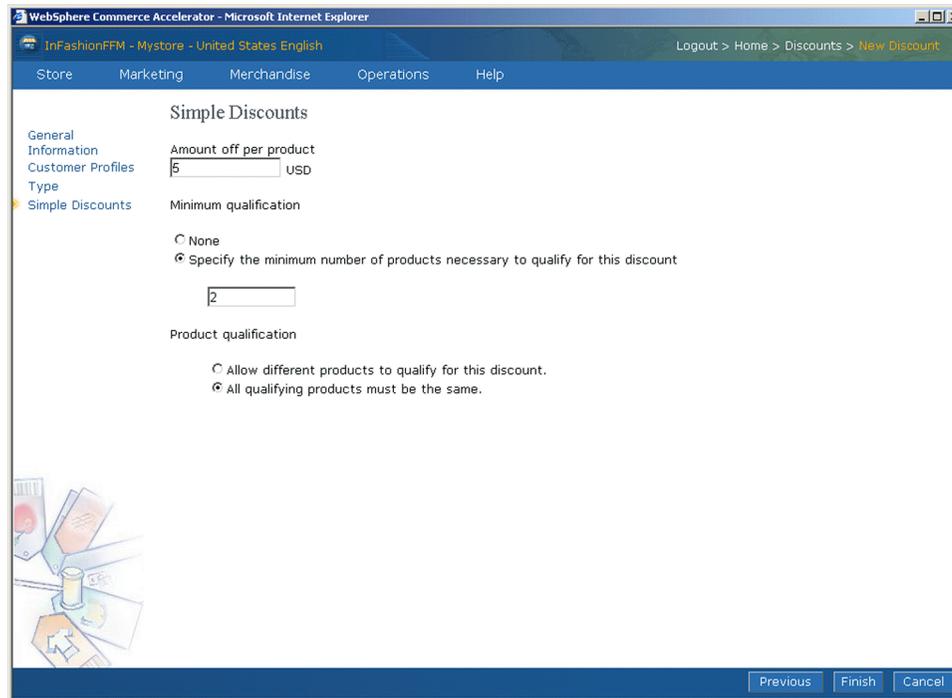


Figure 10-16 Simple discounts

Activating a discount

Next we need to activate the discounts.

- Choose **Merchandise -> Discounts**. The discounts window displays a list of the currently defined discounts.
- Select the check box to the left of the discount that you want to activate. Click **Activate**. The discount will be activated on the production server. See Figure 10-17

Discount name	Status	Start date	End date	Description
<input checked="" type="checkbox"/> IBMWCS54Discount	Active	5/2/02 3:56 PM	Never	Discount of IBMWCS54 testgroup

Figure 10-17 Activate a discount

Changing discount assignment for the products

To display the discount in the order summary page of the sample store we have to add the discount to the product. The steps are:

1. Choose **Merchandise** -> **Products**. A list of products for the store is displayed.
2. Open the change product notebook by selecting the check box to the left of the product that you want to work with and click **Change**.
3. From the left navigation frame, click **Discounts**. The discounts page is displayed.
4. To assign a discount to the product, select the **IBMWCS54Discount** from the available discounts list, and click **Add**.
5. Click **OK**. You will get a message that the discount has been applied successfully.

Display discount in the Order Summary page

1. Launch your store by entering the following:
`https://host_name/webapp/wcs/stores/store_directory/index.jsp`
2. Register as a user of the IBMWCS54 group.
3. Browse the catalog and add some products into the shopping cart. Make sure to add some products including the new discount. Change the quantity from 1 to 2.
4. On the order summary page you will get a discount similar to that shown in Figure 10-18.

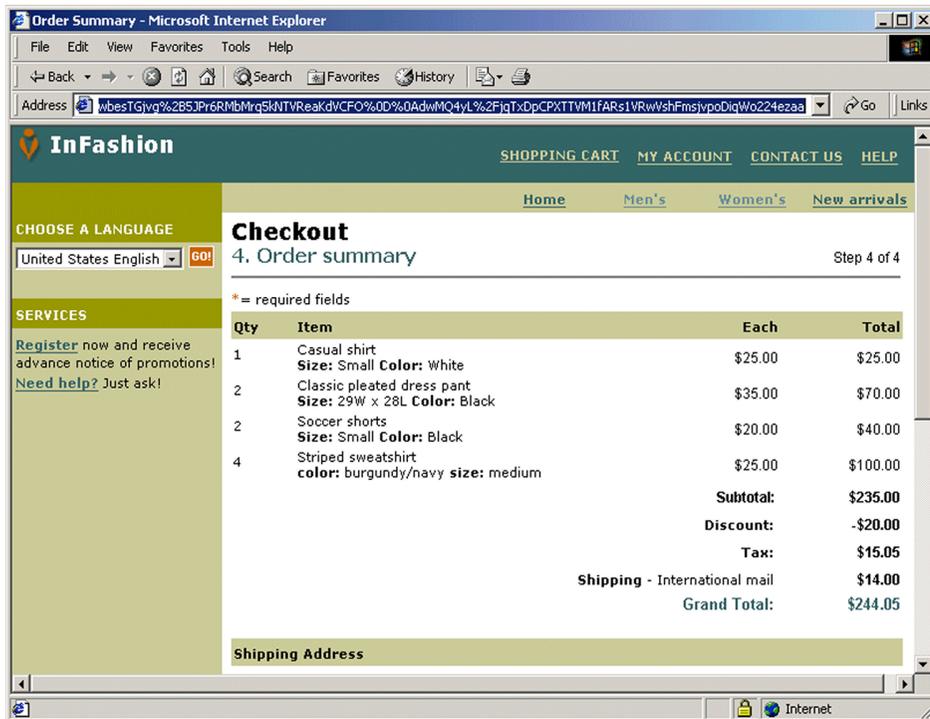


Figure 10-18 Order summary page with simple discount

The customer bought four striped sweatshirts which each cost \$25. The total price of this order item without discount is \$100. You will get a discount of \$20 as shown in Figure 10-18.

Table 10-4 shows how this simple discount is calculated.

Table 10-4 Simple discount

discount type	rule	calculation	result
Amount off per product	When 2 or more striped shirts are chosen, get \$5 off each shirt	$[4 \times (\$25 - \$5)]$	\$80



Messaging customization

In this chapter we look at how to customize the messaging subsystem. Due to the fact that information on how to install and configure MQSeries is a little hard to find in the online documentation, the first thing we present in this chapter is a guide on how to get MQSeries to work with WebSphere Commerce and VisualAge for Java. When that is done, we walk through a customization of the messaging subsystem. This includes creating a new XML and DTD file. We map the inbound XML to command objects. These command objects will be handled by a controller command and several task commands that will perform the task of updating the database with the information received in the inbound xml. Finally we describe the steps needed to be able to test the commands.

11.1 Installing and configuring MQSeries

This section describes how to install MQSeries and perform some basic configuration steps. We used MQSeries Version 5.2.1.

11.1.1 Installing

Install MQSeries choosing custom installation. In the examples in this chapter we used <drive>\MQSeries as installation directory. Make sure that client has been selected as part of the MQSeries install.

11.1.2 Configuring

Perform the following to configure MQSeries

Create a queue manager

1. Open the MQSeries Explorer and create a new queue manager by right-clicking on the folder **Queue Managers**. We use the name QM.DEMO. You can choose any name you like as long as you use it consistently. Specify a dead letter queue as SYSTEM.DEAD.LETTER.QUEUE. See Figure 11-1

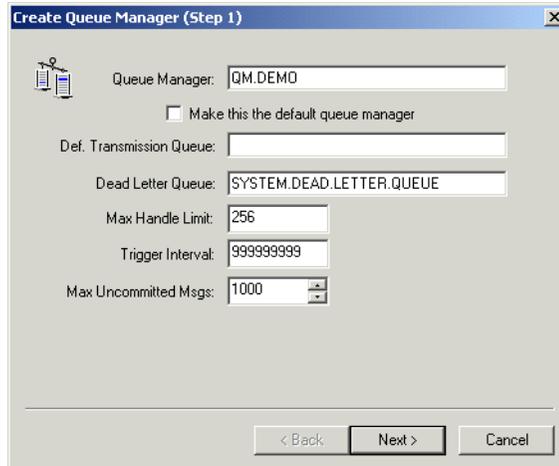


Figure 11-1 Creating MQSeries Queue manager step1

2. Click **Next** twice and check the **Create Server Connection Channel** checkbox. See Figure 11-2

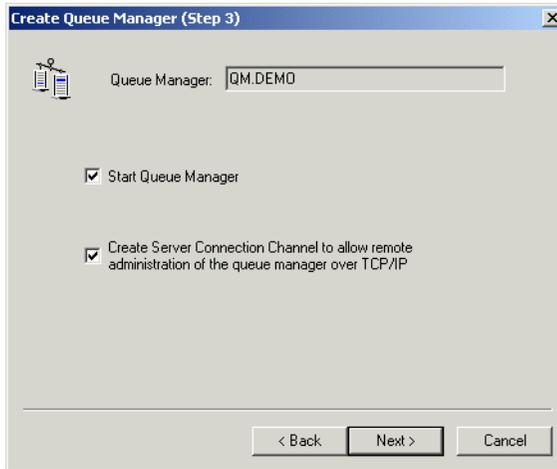


Figure 11-2 Creating MQSeries Queue manager step3

3. Click **Next** and check **Create listener configured for TCP/IP**. The default port is 1414. See Figure 11-3

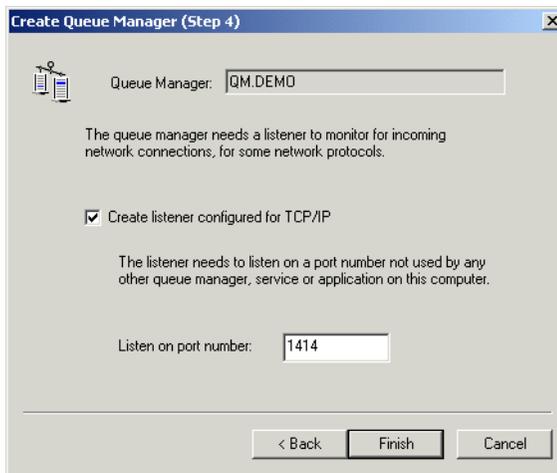


Figure 11-3 MQSeries listener

Define MQSeries queues

You need to define five different queues.

1. Right-click on the **Queues** folder under your queue manager and click **New** -> **Local Queue**. In Queue Name specify wcs_inbound. Click **Ok**. See Figure 11-4

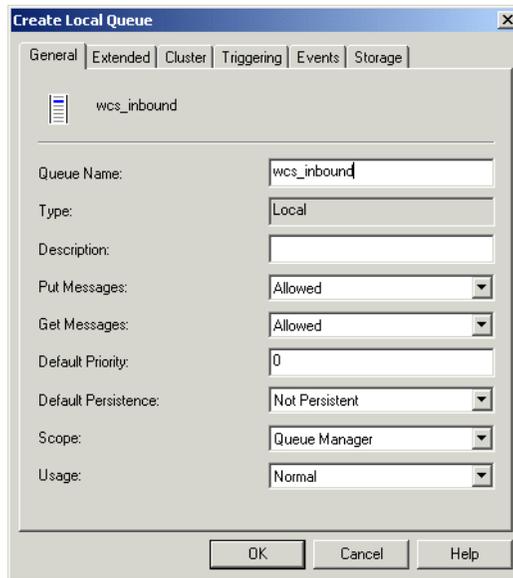


Figure 11-4 Define queues

Create the following five queues:

- ▶ wcs_inbound
- ▶ wcs_inbound_ser
- ▶ wcs_inbound_par
- ▶ wcs_outbound
- ▶ wcs_error

Change coded character set on queue manager

Follow the online documentation on how to change the coded character set to UTF-8 on your queue manager. The details will be in drive:\WebSphere\CommerceServer\web\doc\en_US\tasks\tcvenmsg.htm

MQSeries MA88 product extension pack

Follow the details in the WebSphere Commerce online documentation to install and configure the MA88 product extension pack. See

drive:\WebSphere\CommerceServer\web\doc\en_US\tasks\tcvma88.htm

In the online documentation we used it wrongly points you to the CommerceServer/bin directory. You should instead look for admin.config in the AppServer/bin directory.

Follow the instructions in the online documentation on how to update the WebSphere Application Server classpath variable. See

<drive>\WebSphere\CommerceServer\web\doc\en_US\tasks\tcvupwas.htm

Setup system classpaths and environment variables. See

<drive>\WebSphere\CommerceServer\web\doc\en_US\tasks\tcvcnfjms.htm

In addition to the files described in the WebSphere Commerce documentation you should also add the file connector.jar to the CLASSPATH environment variable.

Copy the files listed below from <drive>\MQSeries\java\lib to the <drive>\WebSphere\AppServer\lib\ext directory:

- ▶ jms.jar
- ▶ connector.jar
- ▶ com.ibm.mq.jar
- ▶ com.ibm.mqjms.jar
- ▶ com.ibm.mqbind.jar
- ▶ jndi.jar
- ▶ jta.jar
- ▶ ldap.jar
- ▶ providerutil.jar

Configure JMS

The next step is to configure JMS using JMSAdmin. See the online documentation

<drive>\WebSphere\CommerceServer\web\doc\en_US\tasks\tcvcnfjms2.htm

We create a file with the queue definitions. We do this since we add the definitions both to WebSphere Application Server and to the WebSphere Test Environment.

The file in Example 11-1 should be saved in <drive>\MQSeries\java\bin. It links the JMS queues to MQSeries. If you have used other names for the queue manager or queues you should change this file accordingly.

Example 11-1 setupJMS.mq

```
define qcf(JMSQueueConnectionFactory) qmanager(QM.DEMO)
alter qcf(JMSQueueConnectionFactory) ccsid(1208)
define q(JMSInboundQueue) queue(wcs_inbound)
define q(JMSSerialInboundQueue) queue(wcs_inbound_ser)
define q(JMSParallelInboundQueue) queue(wcs_inbound_par)
define q(JMSOutboundQueue) queue(wcs_outbound)
define q(JMSErrorQueue) queue(wcs_error)
```

```
alter q(JMSOutboundQueue) targclient(MQ) qmanager(QM.DEMO)
alter q(JMSErrorQueue) targclient(MQ) qmanager(QM.DEMO)
end
```

To execute this configuration file you should enter the following command from the <drive>\MQSeries\java\bin directory. You must have a name server running though, so do not execute this command yet.

```
JMSAdmin -cfg JMSAdmin.config -t -v < setupJMS.mq
```

Stop and restart WebSphere Application Server.

11.2 Enabling the MQ adapter in WebSphere Commerce

You should have completed all the steps in 11.1, “Installing and configuring MQSeries” on page 292 before performing these steps.

11.2.1 TransportAdapter

Launch the WebSphere Commerce Configuration and enable the TransportAdapter component. See Figure 11-5

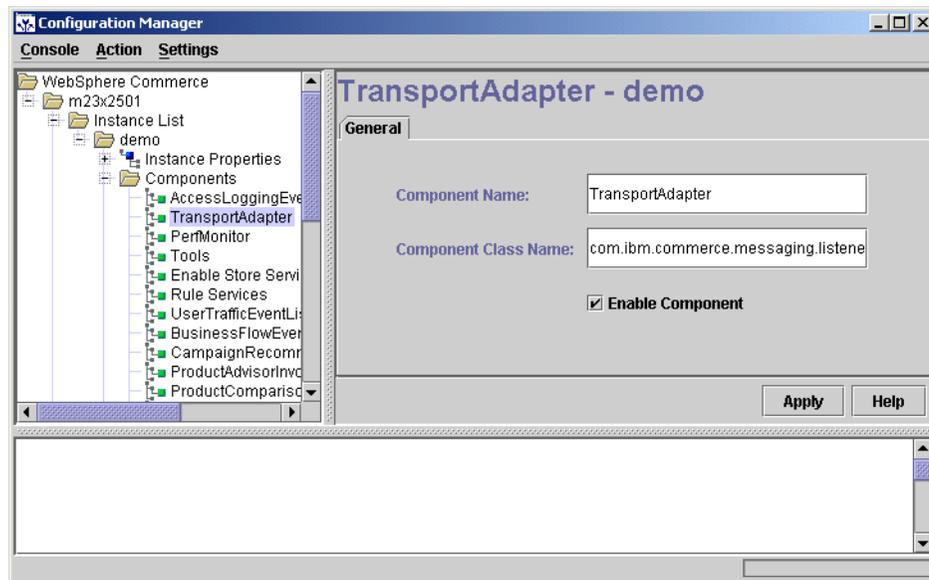


Figure 11-5 Enabling the TransportAdapter component

11.2.2 Log level

When developing commands for the messaging subsystem it is a good idea to raise the loglevel for the messaging and transport components. This makes debugging a lot easier. See Figure 11-6 for an example of how to do this using the WebSphere Commerce Configuration Manager

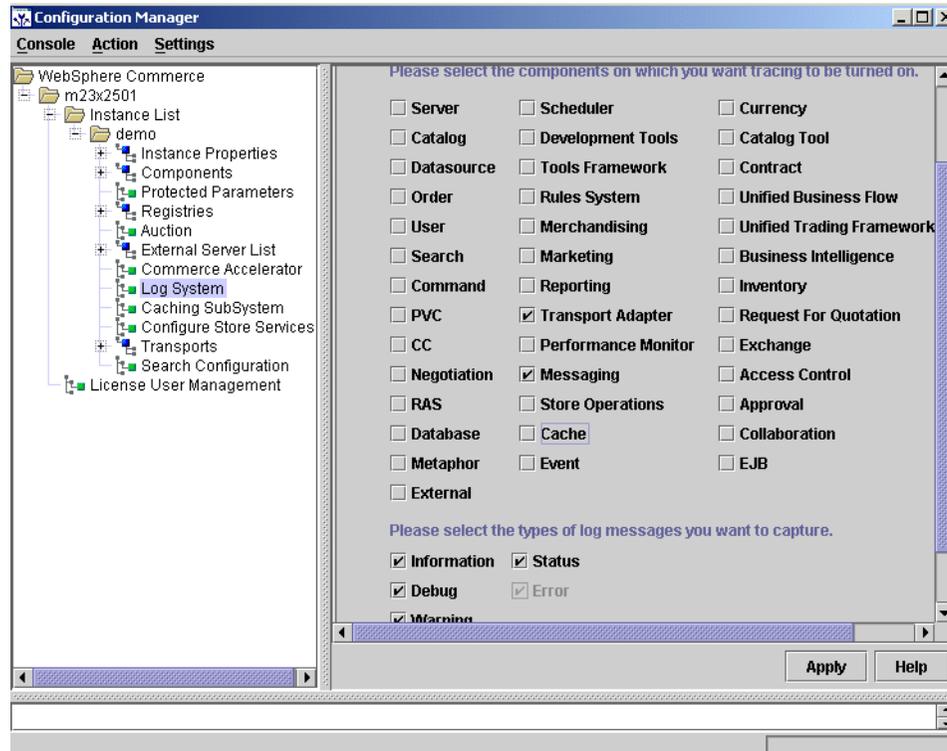


Figure 11-6 Raising the loglevel.

11.3 Enabling the MQ adapter in VisualAge for Java

You should have completed all the steps in 11.1, “Installing and configuring MQSeries” on page 292 before performing these steps.

11.3.1 TransportAdapter

You need to enable the TransportAdapter component in the instance that has been created for VisualAge for Java. You can enable it directly in the XML file for the instance. The file can be found in the following folder:

`drive:\WebSphere\CommerceServerDev\instances\VAJ_Instancename\xml\VAJ_Instancename.xml`

Look for the component TransportAdapter at change the attribute enable to true.

Importing the MA88 product extension

VisualAge for Java needs the MQSeries classes for Java for messaging to work inside the WebSphere Test Environment.

For details on how to do this refer to the Web page at:

<http://www-106.ibm.com/developerworks/ibm/library/it-farrell1/index.html>

Look for the section Adding MQSeries libraries and resources to VisualAge for Java.

If for some reason you cannot view the Web page, then follow these steps:

1. Create a new project. Call it any name you like, for example MA88
2. Import the following jar files into the project. They are all in the `<drive>\MQSeries\java\lib` folder if you followed the installation instructions above. Some of the imported packages will create open editions of existing packages in the *IBM WC Commerce Server 54* project.
 - jms.jar
 - connector.jar
 - com.ibm.mq.jar
 - com.ibm.mqjms.jar
 - com.ibm.mqbind.jar
3. Import the following directories into the project
 - a. `<drive>\MQSeries\java\lib` - uncheck the .java box and in resources you should select only the mqji.properties file.
 - b. `<drive>\MQSeries\java\bin` - uncheck the .java box and in resources you should select only the JMSAdmin.config

Configure JMS

Ensure that the WebSphere Test Environment Persistent Name Server is running and enter the following from the command line in the `<drive>\MQSeries\java\bin` directory:

```
JMSAdmin -cfg JMSAdmin.config -t -v < setupJMS.mq
```

You created the setupJMS.mq file in section 11.1, “Installing and configuring MQSeries” on page 292 above. If you look in the VisualAge for Java console for the Persistent Name Server you will notice that it registers the queue names.

At this stage the Messaging Subsystem should be operational in the WebSphere Test Environment.

11.4 Functional requirements

In WebSphere Commerce there are two ways in which to create new products or make updates to existing products in a store. You can either load the product data using the massloader utility or you can use the WebSphere Commerce Accelerator.

For some companies it is essential to be able to administer their categories and products in only one place - their back-end system. They need to know that when a product is created in the back-end system the commerce site will reflect that change within seconds.

We will create a third method for product creation that will use MQSeries.

11.4.1 Use case

The following is a description of what we want to accomplish.

Basic flow

1. A user on the back-end system will create a product with items and prices and associate the product with a category.
2. Once created the back-end system will compose an XML message that it puts on a MQ queue.
3. The WebSphere Commerce MQ adapter will get the message from the queue, process the data and call the MQProductCreate command.
4. The MQProductCreate command will call the appropriate task commands that will create the product in the WebSphere Commerce database.
5. Users will be able to see the product in the store.

Alternate flow 1

1. A user on the back-end system modifies an already existing product and submits the change.
2. An XML message will be composed that contains only the XML elements that have changed.
3. See steps 3 through 5 in “Basic flow” on page 299

Exception flow

If an error occurs during the load then it will be logged, and an administrator will be able to trace the error. Future versions of the command could implement an optional E-mail notification functionality that will notify the back-end user with the status of the import. However if the error is traced to an XML format that is not well formed according to the DTD then such a notification would not be possible.

If you only want to be notified in case of an error, you can enable the built in WebSphere Commerce error notification. The error notification feature can be configured to send an E-mail to a predefined E-mail address whenever an error occurs. This is a good feature although it would have been better if you could configure it on the component level. If that was possible, you could have one person receiving all E-mails for the messaging component and another person receiving E-mail for other components.

Tip: If you enable WebSphere Commerce error notification when developing in VisualAge for Java, remember to add `activation.jar` and `mail.jar` to your servlet engine's extra classpath.

11.5 Preparing for the MQProductCreate command

We recommend reading the Connectivity and Notification: Online Help that can be downloaded from the WebSphere Commerce Technical Library:

http://www-4.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

In it is a great deal of information on the messaging subsystem and although for this chapter covers all aspects of customizing the messaging subsystem, it is still a good idea to read up on connectivity and notification.

11.5.1 Defining XML and DTD for the command

In this section we define the DTD for the inbound message. We have listed a sample XML file below the DTD. It is largely self explanatory. The only thing that needs special attention are the attributes and attribute values. Both attributes and attribute values are defined on the product. They are then related to items afterwards. The multi language support in attributes is a little tricky. First you create an attribute in the store default language, and then you create an attribute in another language that references the attribute in the store default language.

The same thing is true for attribute values. You have to create reference values in the store default language and then afterwards create values in other languages.

For simplicity we have left out some less important attributes for example the auxiliary descriptions on the product and item descriptions. The commands can easily be modified to implement these attributes.

You should verify that your XML and DTD are well formed. There is a lot of different tools out there to help you do that. One of them is Xmlspy which can be found following this link:

<http://www.xmlspy.com/>

Example 11-2 shows the product creation DTD file we used.

Example 11-2 Create_WCS_Product.dtd

```
<!ELEMENT Create_WCS_Product (ControlArea, DataArea)>
<!ATTLIST Create_WCS_Product
    version CDATA #FIXED "1.0"
>
<!ENTITY % defineNCCCommonModule SYSTEM "NCCCommon.mod">
%defineNCCCommonModule;
<!ELEMENT ControlArea (Verb, Noun, Credentials?)>
<!ELEMENT Verb (#PCDATA)>
<!ATTLIST Verb
    value CDATA #FIXED "Create"
>
<!ELEMENT Noun (#PCDATA)>
<!ATTLIST Noun
    value CDATA #FIXED "WCS_Product"
>
<!ELEMENT Credentials (LogonId, Password)>
<!ELEMENT LogonId (#PCDATA)>
<!ELEMENT Password (#PCDATA)>
<!-- =====DataArea===== -->
<!ELEMENT DataArea (product+)>
<!ELEMENT product (description, categoryrelations*, attributes*, items*)>
<!ATTLIST product
    languageId CDATA #REQUIRED
    ownerorganizationDN CDATA #REQUIRED
    storeentidentifier CDATA #REQUIRED
    markfordelete CDATA #REQUIRED
    mfname CDATA #REQUIRED
    mfpartnumber CDATA #REQUIRED
    partnumber CDATA #REQUIRED
>

<!ELEMENT description EMPTY>
<!ATTLIST description
    available CDATA #REQUIRED
    published CDATA #REQUIRED
```

```

    name CDATA #IMPLIED
    shortdescription CDATA #IMPLIED
    longdescription CDATA #IMPLIED
    thumbnail CDATA #IMPLIED
    fullimage CDATA #IMPLIED
    keyword CDATA #IMPLIED
    availabilitydate CDATA #IMPLIED
  >
<!ELEMENT name (#PCDATA)>
<!ELEMENT published (#PCDATA)>
<!ELEMENT categoryrelations (categoryrelation+)>
<!ELEMENT categoryrelation EMPTY>
<!ATTLIST categoryrelation
  action CDATA #REQUIRED
  categoryidentifier CDATA #REQUIRED
  catalogidentifier CDATA #REQUIRED
  sequence CDATA #REQUIRED
>
<!ELEMENT attributes (attribute+)>
<!ELEMENT attribute (attrvalue+)>
<!ATTLIST attribute
  referencename CDATA #IMPLIED
  name CDATA #REQUIRED
  sequence CDATA #IMPLIED
  type CDATA #IMPLIED
  action CDATA #IMPLIED
>
<!ELEMENT items (item+)>
<!ELEMENT item (description, attrvalues*)>
<!ATTLIST item
  markfordelete CDATA #IMPLIED
  mfpartnumber CDATA #IMPLIED
  partnumber CDATA #REQUIRED
>
<!ELEMENT attrvalues (itemattrvalue+)>
<!ELEMENT attrvalue EMPTY>
<!ATTLIST attrvalue

  referencevalue CDATA #REQUIRED
  value CDATA #REQUIRED
  sequence CDATA #IMPLIED
  action CDATA #IMPLIED
>
<!ELEMENT itemattrvalue EMPTY>
<!ATTLIST itemattrvalue
  referencevalue CDATA #REQUIRED
  attributename CDATA #REQUIRED

```

>

The XML in Example 11-3 defines a product with a description, one or more relations to existing categories, attributes with attribute values and items belonging to the product with their descriptions and relations to attribute values. Notice that no internal WebSphere Commerce generated ids are used. This means that the backend system does not have to store the ids.

Example 11-3 Create_WCS_Product_en_US.xml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Create_WCS_Product SYSTEM "Create_WCS_Product.dtd">
<Create_WCS_Product>
  <ControlArea>
    <Verb value="Create" />
    <Noun value="WCS_Product" />
  </ControlArea>
  <DataArea>
    <product languageId="-1" ownerorganizationDN="MyOrganization"
storeentidentifier="webfashion" markfordelete="0" mfname="WebFashion Inc."
mfpartnumber="1234-123-400" partnumber="1234-123-400">
      <description
        available="0"
        published="1"
        name="MyOwnProduct"
        shortdescription="My Second Product"
        longdescription="This is the long description of My Second
Product, and I can add more descriptive information for the product and just
load it through MQ
        thumbnail="images/1234-123-400_small.jpg"
        fullimage="images/1234-123-400_big.jpg"
        keyword=""
        availabilitydate=""
      />
      <categoryrelations>
        <categoryrelation categoryidentifier="Outerwear"
catalogidentifier="WebFashion" sequence="0" action="create"/>
      </categoryrelations>
      <attributes>
        <attribute type="STRING" name="Color" referencename="Color"
sequence="0">
          <attrvalue referencevalue="Red" value="Red" sequence="0"/>
          <attrvalue referencevalue="Blue" value="Blue" sequence="0"/>
        </attribute>
        <attribute type="STRING" name="Size" referencename="Size"
sequence="0">
          <attrvalue referencevalue="S" value="S" sequence="0"/>
          <attrvalue referencevalue="M" value="M" sequence="0"/>
        </attribute>
      </attributes>
    </product>
  </DataArea>
</Create_WCS_Product>
```

```

        </attribute>
    </attributes>
    <items>
        <item markfordelate="0" mfpartmentnumber="1234-123-400-sku1"
partnumber="1234-123-400-sku1">
            <description
                available="0"
                published="1"
                name="MyOwnProduct sku1"
                shortdescription="My Second Product sku1"
                longdescription="This is the long description of My Second
Product sku1.
                "
                thumbnail="images/1234-123-400_small.jpg"
                fullimage="images/1234-123-400_big.jpg"
                keyword=""
                availabilitydate=""
            />
            <attrvalues>
                <itemattrvalue attributename="Color" referencevalue="Red"
/>
                <itemattrvalue attributename="Size" referencevalue="S" />
            </attrvalues>
        </item>
        <item markfordelate="0" mfpartmentnumber="1234-123-400-sku2"
partnumber="1234-123-400-sku2">
            <description
                available="0"
                published="1"
                name="MyOwnProduct sku2"
                shortdescription="My Second Product sku2"
                longdescription="This is the long description of My Second
Product sku2.
                "
                thumbnail="images/1234-123-400_small.jpg"
                fullimage="images/1234-123-400_big.jpg"
                keyword=""
                availabilitydate=""
            />
            <attrvalues>
                <itemattrvalue attributename="Color" referencevalue="Blue"
/>
                <itemattrvalue attributename="Size" referencevalue="S" />
            </attrvalues>
        </item>
    </items>
</product>
</DataArea>
</Create_WCS_Product>

```

It is necessary to create an XML file for every language that you wish to support in the store. The attributes called `referencename` and `referencevalue` should be the same as the corresponding names and values in the XML file for the store default language. And the attribute called `attributename` on the `itemattrvalue` element should still point to the names of the attributes in the store default language. There is a German language sample of the XML in Example 11-3 in the additional material that can be downloaded for our redbook. See Appendix B, “Additional material” on page 409 for details on downloading this material.

11.5.2 Registering the DTD in WebSphere Commerce

It is necessary to register the DTD in the list of known inbound message DTD files. Otherwise WebSphere Commerce will not know what to do with the XML. It needs to know which command to call. If you forget to register the DTD or if the message is not well-formed you will get the following error message in the `ecmsg*.log`:

```
Message Text: ERROR CMN8124E New inbound message command is not customized.
```

You can register the DTD in the Configuration Manager using **Instance Properties -> Messaging -> Inbound Message DTD Files**.

Copy the DTD file to the following directory:

```
drive:\WebSphere\CommerceServer\xml\messaging
```

11.5.3 Creating mapping between the XML and the command

When the inbound XML file is taken off of the queue by the MQ listener it has to go through a message mapper to determine what has to be done with the message. By default WebSphere Commerce has two message mappers defined: `WCS.INTEGRATION` for parsing XML and `NC.LEGACY` for parsing back-end integration legacy messages. The message mapper has to verify and convert the inbound message to a command property object so it can be used by a WebSphere Commerce command. To verify the inbound message it uses the DTD described in Example 11-2 on page 301, and to convert the XML it uses a template XML file. For the built-in messages the file is called `sys_template.xml` and for custom messages it is `user_template.xml`. This can be customized in the Configuration Manager.

To convert the message to a command property object you must create a template document in the `user_template.xml` file.

To get an idea of how to create the tags you can look for inspiration in the `sys_template.xml` file or you can go to the following Web page to learn about the XML Path Language (XPath):

<http://www.w3.org/TR/xpath>

You should think object oriented when creating the tags. Consider if a tag in the inbound message is a field or an object and if these fields or objects should be put in a container for example a Vector. In this example <categoryrelation> is a container object that holds attributes, therefore it should be defined as XPathType='Vector' in the user_template.xml. See Example 11-4 for the user template we used.

Example 11-4 user_template.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE ECTemplate SYSTEM 'ec_template.dtd' >
<ECTemplate>
<TemplateDocument>
<DocumentType version='1.0'>Create_WCS_Product</DocumentType>
<StartElement>Create_WCS_Product</StartElement>
<TemplateTagName>Create_WCS_ProductMap</TemplateTagName>
<CommandMapping>
<Command CommandName='MQProductCreate' />
</CommandMapping>
</TemplateDocument>
<TemplateTag name='Create_WCS_ProductMap'>
  <!-- Execution Environment -->
  <Tag XPath='ControlArea/Credentials/LogonId' Field='logonId'
FieldInfo='CONTROL' />
  <Tag XPath='ControlArea/Credentials/Password' Field='logonPassword'
FieldInfo='CONTROL' />

  <!-- Command Parameters -->
  <Tag XPath='DataArea/product' Field='product' XPathType='VECTOR' />
  <Tag XPath='DataArea/product@languageId' Field='languageId' />
  <Tag XPath='DataArea/product@ownerorganizationDN'
Field='ownerorganizationDN' />
  <Tag XPath='DataArea/product@storeentidentifier' Field='storeentIdentifier'
/>
  <Tag XPath='DataArea/product@markfordelete' Field='markForDelete' />
  <Tag XPath='DataArea/product@mfname' Field='manufacturerName' />
  <Tag XPath='DataArea/product@mfpnumber' Field='manufacturerPartNumber'
/>
  <Tag XPath='DataArea/product@partnumber' Field='partnumber' />

  <Tag XPath='DataArea/product/description' Field='descriptionVector'
XPathType='VECTOR' />
  <Tag XPath='DataArea/product/description@available' Field='available' />
  <Tag XPath='DataArea/product/description@published' Field='published' />
  <Tag XPath='DataArea/product/description@name' Field='name' />
  <Tag XPath='DataArea/product/description@shortdescription'
Field='shortdescription' />
```

```

    <Tag XPath='DataArea/product/description@longdescription'
Field='longdescription' />
    <Tag XPath='DataArea/product/description@thumbnail' Field='thumbnail' />
    <Tag XPath='DataArea/product/description@fullimage' Field='fullimage' />
    <Tag XPath='DataArea/product/description@keyword' Field='keyword' />
    <Tag XPath='DataArea/product/description@availabilitydate'
Field='availabilitydate' />

    <Tag XPath='DataArea/product/categoryrelations/categoryrelation'
Field='categoryrelationVector' XPathType='VECTOR' />
    <Tag
XPath='DataArea/product/categoryrelations/categoryrelation@categoryidentifier'
Field='categoryidentifier' />
    <Tag
XPath='DataArea/product/categoryrelations/categoryrelation@catalogidentifier'
Field='catalogidentifier' />
    <Tag XPath='DataArea/product/categoryrelations/categoryrelation@sequence'
Field='sequence' />
    <Tag XPath='DataArea/product/categoryrelations/categoryrelation@action'
Field='action' />

    <Tag XPath='DataArea/product/attributes/attribute' Field='attributeVector'
XPathType='VECTOR' />
    <Tag XPath='DataArea/product/attributes/attribute@name' Field='name' />
    <Tag XPath='DataArea/product/attributes/attribute@sequence'
Field='sequence' />
    <Tag XPath='DataArea/product/attributes/attribute@type' Field='type' />
    <Tag XPath='DataArea/product/attributes/attribute@referencename'
Field='reference' />
    <Tag XPath='DataArea/product/attributes/attribute/attrvalue'
Field='attrValueVector' XPathType='VECTOR' />
    <Tag XPath='DataArea/product/attributes/attribute/attrvalue@value'
Field='value' />
    <Tag XPath='DataArea/product/attributes/attribute/attrvalue@sequence'
Field='sequence' />
    <Tag XPath='DataArea/product/attributes/attribute/attrvalue@referencevalue'
Field='referencevalue' />

    <Tag XPath='DataArea/product/items/item' Field='itemVector'
XPathType='VECTOR' />
    <Tag XPath='DataArea/product/items/item@markfordelete'
Field='markForDelete' />
    <Tag XPath='DataArea/product/items/item@mfpartnumber'
Field='manufacturerPartnumber' />
    <Tag XPath='DataArea/product/items/item@partnumber' Field='partnumber' />

    <Tag XPath='DataArea/product/items/item/description'
Field='itemDescriptionVector' XPathType='VECTOR' />

```

```

    <Tag XPath='DataArea/product/items/item/description@available'
Field='available' />
    <Tag XPath='DataArea/product/items/item/description@published'
Field='published' />
    <Tag XPath='DataArea/product/items/item/description@name' Field='name' />
    <Tag XPath='DataArea/product/items/item/description@shortdescription'
Field='shortdescription' />
    <Tag XPath='DataArea/product/items/item/description@longdescription'
Field='longdescription' />
    <Tag XPath='DataArea/product/items/item/description@thumbnail'
Field='thumbnail' />
    <Tag XPath='DataArea/product/items/item/description@fullimage'
Field='fullimage' />
    <Tag XPath='DataArea/product/items/item/description@keyword'
Field='keyword' />
    <Tag XPath='DataArea/product/items/item/description@availabilitydate'
Field='availabilitydate' />

    <Tag XPath='DataArea/product/items/item/attrvalues/itemattrvalue'
Field='itemAttrValueVector' XPathType='VECTOR' />
    <Tag
XPath='DataArea/product/items/item/attrvalues/itemattrvalue@attributename'
Field='attributename' />
    <Tag
XPath='DataArea/product/items/item/attrvalues/itemattrvalue@referencevalue'
Field='referencevalue' />

</TemplateTag>
</ECTemplate>

```

The user_template.xml file should be placed in the following directory:

```
<drive>:\WebSphere\CommerceServer\xml\messaging
```

You will need to restart the application server before the changes to the template will be recognized..

11.5.4 Registering the command in WebSphere Commerce

The commands need to be registered in the URLREG and in the CMDREG database tables, if you are not using the defaultCommandClassName in the interface. In this example we will not register the command in CMDREG since the command implementation will be defined in the interface. Example 11-5 shows the SQL used to register our command.

Example 11-5 Register MQProductCreate in cmdreg

```
db2 insert into urlreg (url, storeent_id, interfacename,https, internal) values
('MQProductCreate', 0,
'com.ibm.commerce.wc54handbook.commands.MQProductCreateCmd', 0, 0)
```

11.6 Creating the commands

Now that everything has been setup it is time to write the actual commands. From the use case we know what is supposed to be done.

We modularize the command into one controller command and five task commands. They are:

- ▶ MQProductCreateCmd
 - MQProductDescriptionsCmd
 - MQProductCategoryRelationsCmd
 - MQProductAttributesCmd
 - MQProductAttributeValuesCmd
 - MQProductItemsCmd

It makes sense to separate the logic into different commands since it enables us to use small parts of the functionality in other appropriate places. It also adds to the readability of the code.

We walk through the writing of the controller command and one of the task commands in this chapter. The task commands do different things, but structurally they are alike.

11.6.1 Creating the MQProductCreateCmd interface

We will create a simple interface for the command. We name it:

```
com.ibm.commerce.wc54handbook.commands.MQProductCreateCmd
```

The interface should extend:

```
com.ibm.commerce.command.ControllerCommand
```

Create a final static field `defaultCommandClassName` and set it to the full classname of our implementation class.

```
String defaultCommandClassName =
"com.ibm.commerce.wc54handbook.commands.MQProductCreateCmdImpl";
```

11.6.2 Creating the MQProductCreate command impl class

This command will be a standard controller command that extends from:

```
com.ibm.commerce.command.ControllerCommandImpl
```

and it should implement the interface we created above.

Setting the request properties

When the MQProductCreate command is invoked the first thing that happens is that its `setRequestProperties` method is called. It takes a `TypedProperty` as an argument. All the variables that we defined in the `user_template.xml` file are in this `typedproperty` object. Contained in the `typedproperty` object are elements for each `<product>` in the inbound xml. We create an instance variable called `productVector` - its type should be `java.util.Vector`. We store the products in this vector. Example 11-6 shows the Java code

Example 11-6 setRequestProperties

```
public void setRequestProperties(TypedProperty typedproperty)
    throws com.ibm.commerce.exception.ECException {
    String methodName = "setRequestProperties";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    super.requestProperties = typedproperty;
    try {
        setProductVector((Vector) super.requestProperties.get("product"));
    } catch (Exception exception) {
        setProductVector(null);
    }
    ECTrace.trace(
        ECTraceIdentifiers.COMPONENT_EXTERN,
        CLASSNAME,
        methodName,
        "Number of products in inbound XML = " + getProductVector().size());
    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}
```

All tracing and logging information is sent to `COMPONENT_EXTERN`, so you should enable that component in the Configuration Manager before testing the command.

Tip: When debugging an MQ command be careful not to click stop. It will most likely freeze your servlet engine. Always run the command to the end.

performExecute

The performExecute method is actually called before setRequestProperties but since we call super.performExecute as the first thing in our performExecute, setRequestProperties is called before any of the business logic in performExecute is run. Example 11-7 shows the Java code.

Example 11-7 performExecute()

```
public void performExecute() throws ECEException {
    String methodName = "performExecute";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    super.performExecute();

    // call doProcess() for every product in the productVector
    for (int j = 0; j < getProductVector().size(); j++) {
        TypedProperty aProduct =
            (TypedProperty) getProductVector().elementAt(j);
        setRequestPropertiesPerProduct(aProduct);
        if (!doProcess(super.requestProperties))
            throw new ECAApplicationException(
                ECTraceIdentifiers.ERR_DO_PROCESS,
                getClass().getName(),
                "performExecute");
        cleanUp();
    }
    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}
```

setRequestPropertiesPerProduct

For each product we need to get the data it contains. We do that in this method. We create Vectors for each of the sub elements the vectors will contain TypedProperty objects for each instance of that sub element. Example 11-8 shows the Java code

Example 11-8 setRequestPropertiesPerProduct

```
private void setRequestPropertiesPerProduct(TypedProperty typedproperty) throws
com.ibm.commerce.exception.ECEException {
    String methodName = "setRequestPropertiesPerProduct";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    setLanguageId(typedproperty.getInteger("languageId", null));
    setOwnerOrganizationDN(
        typedproperty.getString("ownerorganizationDN", null));
    setStoreentIdentifier(typedproperty.getString("storeentIdentifier", null));
    setPartnumber(typedproperty.getString("partnumber", null));
}
```

```

setPublished(typedproperty.getInteger("published", null));
setMarkForDelete(typedproperty.getInteger("markForDelete", null));
setManufacturerName(typedproperty.getString("manufacturerName", ""));
setManufacturerPartnumber(
    typedproperty.getString("manufacturerPartnumber", "");
setDescriptionVector(
    (Vector) typedproperty.get("descriptionVector", null));
setCategoryrelationVector(
    (Vector) typedproperty.get("categoryrelationVector", null));
setAttributeVector((Vector) typedproperty.get("attributeVector", null));
setItemVector((Vector) typedproperty.get("itemVector", null));

ETrace.exit(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}

```

We work with the attribute values when we work with each item.

doProcess

The doProcess method reflects the conditions in the use case.

1. If it is a new product then all the elements must exist in the XML files and thus in the typedproperty object for the product.
2. If the product exists then it should be possible only to update for example a single description or update only categoryrelations. Thus we must check if the different vectors contain anything.

Example 11-9 shows the Java code

Example 11-9 doProcess

```

public boolean doProcess(TypedProperty typedproperty) throws ECEException {
    String methodName = "doProcess";
    ETrace.entry(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
    getCommandContext().setStoreId(getProductStoreId());

    if (isNewProduct()) {
        doCreateProduct();
        doProcessDescription();
        doProcessCategoryRelations();
        doProcessAttributes();
        doProcessItems();
    } else {
        doUpdateProduct();
        if (getDescriptionVector() != null && getDescriptionVector().size() > 0)
            doProcessDescription();
        if (getCategoryrelationVector() != null
            && getCategoryrelationVector().size() > 0)
            doProcessCategoryRelations();
    }
}

```


doCreateProduct

Luckily we do not have to do all the work ourselves when creating a product. WebSphere Commerce comes with a command that will do the job. This command is what makes it possible to create products in WebSphere Commerce Accelerator . The command is called `CatalogEntryAddCmd` and you can read about it on the following page of the WebSphere Commerce Version 5.4 Online Help:

```
<drive>\WebSphere\CommerceServer\web\doc\en_US\refs\rcacenad.htm
```

We check for null on the variables that are required and create a typedproperty object that will work as request property for the `CatalogEntryAddCmd`.

We also need to set the storeId on the `CommandContext`. We create a method called `getProductStoreId` for that purpose. Additionally we create a method called `getProductOwner`. We will use that as the `memberId` on the product - the owner of the product. Example 11-11 shows the Java code.

Example 11-11 doCreateProduct

```
private void doCreateProduct() throws ECEException {
    String methodName = "doCreateProduct";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    try {
        TypedProperty catProp = new TypedProperty();
        catProp.put("partnumber", getPartnumber());
        catProp.put("catenttypeId", "ProductBean");
        catProp.put("buyable", new Integer(0));
        catProp.put("markForDelete", getMarkForDelete());
        catProp.put("mfName", getManufacturerName());
        catProp.put("mfPartnumber", getManufacturerPartnumber());
        CatalogEntryAddCmd catalogentryadd = null;
        catalogentryadd =
            (CatalogEntryAddCmd) CommandFactory.createCommand(
                "com.ibm.commerce.catalogmanagement.commands.CatalogEntryAddCmd",
                getProductStoreId());
        catalogentryadd.setCommandContext(getCommandContext());
        catalogentryadd.setMemberId(getProductOwner());
        catalogentryadd.setRequestProperties(catProp);
        catalogentryadd.execute();
        setProductId(catalogentryadd.getCatentryId());
    } catch (ECEException ece) {
        if (ece.getThrowable() instanceof javax.ejb.DuplicateKeyException) {
            throw new ECAApplicationException(
                ECTraceIdentifiers.COMPONENT_EXTERN,
                CLASSNAME,
                methodName,
                ECTraceIdentifiers.COMPONENT_EXTERN.generateMsgParms(
```

```

        "Trying to create product that already exists - possible
        MARKFORDELETE = 1."));
    }
    throw ece;
}
ETrace.entry(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}

```

If the `CatalogEntryAdd` command throws an exception, we want to know if its a `DuplicateKeyException` since that most likely means the product has `markfordelete=1`, we wont be able to update a product that has `markfordelete=1`

Tip: In addition to the `markfordelete` attribute, you could create a manual override attribute that if set will delete the catalog entry at once instead of having to wait for DB clean to get around to do it. This is helpful if you have loaded a product by mistake.

The `doUpdateProduct` method works similar to the `doCreateProduct` except it uses a different WebSphere Commerce command and it needs to be supplied with the catalog entry reference number.

We now have either created or updated the actual product in the database. But there is still a lot to do. The different task commands will all do a small part of the job. We will focus on the `doProcessAttributes()` method. You will be able to examine the rest of the task commands in the additional material that can be downloaded for our redbook. See Appendix B, “Additional material” on page 409 for details on downloading this material.

doProcessAttributes

Creating attributes and attribute values for products in WebSphere Commerce can be a troublesome task. You will get an idea of how complicated the attribute model is, when you create products with attributes in the WebSphere Commerce Accelerator.

We have isolated the handling of attributes in a task command called `MQProductAttributesCmd`. Example 11-12 shows the Java code.

Example 11-12 doProcessAttributes

```

private void doProcessAttributes() throws ECEException {
    String methodName = "doProcessAttributes";
    ETrace.entry(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);

    MQProductAttributesCmd attributeCmd = null;
    attributeCmd = (MQProductAttributesCmd) CommandFactory.createCommand

```

```

        ("com.ibm.commerce.wc54handbook.commands.MQProductAttributesCmd",
         getCommandContext().getStoreId());
attributeCmd.setCommandContext(getCommandContext());
attributeCmd.setProductId(getProductId());
attributeCmd.setStoreDefaultLang(getStoreDefaultLanguageId());
attributeCmd.setLanguageId(getLanguageId());
attributeCmd.setAttributes(getAttributeVector());
attributeCmd.execute();

ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}

```

11.6.3 Creating the MQProductAttributesCmd interface

This interface will be named:

```
com.ibm.commerce.wc54handbook.commands.MQProductCreateCmd
```

The interface should extend:

```
com.ibm.commerce.command.TaskCommand
```

Create a final static field `defaultCommandClassName` and set it to the full classname of our implementation class.

```
String defaultCommandClassName =
"com.ibm.commerce.wc54handbook.commands.MQProductAttributesCmdImpl";
```

Methods

The interface will need the following methods:

- ▶ `setAttributes(Vector)`
- ▶ `setLanguageId(Integer)`
- ▶ `setProductId(Long)`
- ▶ `setStoreDefaultLang(Integer)`

11.6.4 Creating the MQProductAttributesCmdImpl class

As mentioned above the `MQProductCreateCmd` uses five different custom task commands. We will go through only one of them here since they are all quite similar. `MQProductDescriptionCmd` and `MQProductCategoryRelationsCmd` are both very simple which is why we don't walk through them here. You will be able to look through them by downloading the additional material for our redbook. See Appendix B, "Additional material" on page 409 for details on downloading this material.

validateParameters

We want to make sure that all the required parameters have been set. So we check for them and throw an `EApplicationException` if they are null. Example 11-13 shows the Java code.

Example 11-13 `validateParameters`

```
public void validateParameters()
    throws com.ibm.commerce.exception.ECException {
    String methodName = "validateParameters";
    ETrace.entry(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
    super.validateParameters();

    if (getProductId() == null) {
        throw new EApplicationException(
            EMessage._ERR_MISSING_PARMS,
            CLASSNAME,
            methodName,
            EMessageHelper.generateMsgParms("productId"));
    }
    if (getAttributes() == null) {
        throw new EApplicationException(
            EMessage._ERR_MISSING_PARMS,
            CLASSNAME,
            methodName,
            EMessageHelper.generateMsgParms("attributes"));
    }
    if (getStoreDefaultLang() == null) {
        throw new EApplicationException(
            EMessage._ERR_MISSING_PARMS,
            CLASSNAME,
            methodName,
            EMessageHelper.generateMsgParms("storeDefaultLang"));
    }
    if (getLanguageId() == null) {
        throw new EApplicationException(
            EMessage._ERR_MISSING_PARMS,
            CLASSNAME,
            methodName,
            EMessageHelper.generateMsgParms("languageId"));
    }

    ETrace.exit(ETraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}
```

performExecute

When the parameters have been validated its time to loop through the attributes that have been supplied in a Vector with the setAttributes method. Each attribute in the Vector is a TypedProperty object. Example 11-14 shows the Java code.

Example 11-14 performExecute

```
public void performExecute() throws com.ibm.commerce.exception.ECException {
    String methodName = "performExecute";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
    super.performExecute();

    for (int j = 0; j < getAttributes().size(); j++) {
        TypedProperty anAttribute = (TypedProperty)
getAttributes().elementAt(j);

        validateAttribute(anAttribute);
        ECTrace.trace(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME,
methodName, "Attribute validated.");

        String name = anAttribute.getString("name");
        String referencename = anAttribute.getString("reference", null);
        String type = anAttribute.getString("type", null);
        Vector attributeValues = (Vector) anAttribute.get("attrValueVector",
null);

        if (isNewAttribute(name, getLanguageId()) &&
(getLanguageId().equals(getStoreDefaultLang()))) {
            createAttribute(anAttribute);
        } else {
            updateAttribute(anAttribute);
        }

        if (attributeValues != null) {
            MQProductAttributeValuesCmd attributeValuesCmd = null;
            attributeValuesCmd = (MQProductAttributeValuesCmd)
CommandFactory.createCommand("com.ibm.commerce.wc54handbook.commands.MQProductA
ttributeValuesCmd", getCommandContext().getStoreId());
            attributeValuesCmd.setCommandContext(getCommandContext());
            attributeValuesCmd.setProductId(getProductId());
            attributeValuesCmd.setStoreDefaultLang(getStoreDefaultLang());
            attributeValuesCmd.setAttributeValues(attributeValues);
            attributeValuesCmd.setAttributeId(getAttributeId());
            attributeValuesCmd.setAttrType(type);
            attributeValuesCmd.setLanguageId(getLanguageId());
            attributeValuesCmd.execute();
        }
    }
}
```

```
    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}
```

Three things take place in this method. We validate every attribute with the method `validateAttribute`. See Example 11-15. We simply test if the parameters that needs validation can be extracted from the `TypedProperty` object in the correct type.

Now that the attribute has been validated we need to find out whether it is a new or an existing attribute that we are about to load. We do that with a method called `isNewAttribute`. See Example 11-16. It takes the name and the `languageId` of the attribute and simply tries to get an instance of an `AttributeAccessBean`. A `FinderException` tells us that it is not an existing attribute.

Example 11-15 validateAttribute

```
public void validateAttribute(
    com.ibm.commerce.datatype.TypedProperty attribute)
    throws ECEException {
    String methodName = "validateAttribute";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
    try {
        attribute.getString("name");
        attribute.getDouble("sequence", new Double(0D));
    } catch (ParameterNotFoundException pnfe) {
        throw new ECAApplicationException(
            ECTraceIdentifiers.ERR_CMD_MISSING_PARAM,
            CLASSNAME,
            methodName,
            ECTraceIdentifiers.generateMsgParms(pnfe.getParamName()));
    }
    ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
}
```

Example 11-16 isNewAttribute

```
private boolean isNewAttribute(String name, Integer languageId)
    throws ECEException {
    String methodName = "isNewAttribute";
    ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME, methodName);
    try {
        AttributeAccessBean attributeAB = new AttributeAccessBean();
        attributeAB.findByNameAndCatalogEntryAndLanguage(
            name,
            getProductId(),
            languageId);
        ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME,
            methodName);
    }
}
```

```

        return false;
    } catch (javax.ejb.FinderException fe) {
        ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN, CLASSNAME,
methodName);
        return true;
    } catch (Exception e) {
        throw new ECApplcationException(
            ECTraceIdentifiers.ERR_GENERIC,
            CLASSNAME,
            methodName,
            ECTraceIdentifiers.generateMsgParms(
                "Error checking attribute. Name: "
                + name
                + " LanguageId: "
                + languageId
                + " ProductId: "
                + getProductId()));
    }
}

```

We now have a validated attribute and we know if its a new or existing attribute. But even though `isNewAttribute` would return true, we would still have to call `updateAttribute` if the attribute to be created is not in the store default language. You do not create new attributes in other languages than the store default, instead you update the attribute in the store default language with a new `languageId` and it will automatically know that an attribute entry in the specific language should be created. That is why we test if `languageId` and store default `languageId` are equal. If they are equal, then we should use `createAttribute` if its a new attribute, and if they are not equal then we should use `updateAttribute`.

createAttribute and updateAttribute

The two methods simply executes the corresponding WebSphere Commerce commands for their purpose. Its important to be aware of when to use create and when to use update for an attribute.

11.7 Testing the MQProductCreate command

We used the WebFashion store as the target of our product imports. Since we have not implemented import of prices through our command you should be aware of the following:

- ▶ If you want to complete a shopping flow with the product you will have to import prices for the product and items manually, or create them in the WebSphere Commerce Accelerator when the products have been loaded.

- ▶ If you do not want to complete a shopping flow but just want to see the product displayed in CategoryDisplay you will have to modify two JSP files. For instructions on how to do that read below in 11.7.2, “Modifying JSP files” on page 321.

Make sure that you have disabled the WebSphere Commerce cache or enabled cache triggers.

11.7.1 Deploying the commands

Export the commands from VisualAge for Java and follow the steps outlined in the *IBM WebSphere Commerce Programmer's Guide Version 5.4* on what needs to be done to the exported jar to complete the deployment.

The jar should be placed in the WEB-INF/lib directory in the stores webapp folder for example:

```
<drive>\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\wc  
stores.war\WEB-INF\lib
```

11.7.2 Modifying JSP files

To be able to view the imported products in the catalog you have to modify some of the JSP files in WebFashion

subcategory.jsp

In WebFashion product beans actually have prices which is a bit peculiar since a product bean is only a template for a group of items. A product is in itself not a fully resolved SKU. Since the product beans have prices in WebFashion they have chosen to show the prices on the product list in CategoryDisplay. In order for subcategory.jsp not to fail when it tries to get the price on the imported product you need to write a try-catch around that section in the code as shown in Example 11-17

Example 11-17 subcategory.jsp

```
for (int i = 0; i < products.length; ++i)  
{  
    product = products[i];  
    try {  
        com.ibm.commerce.price.beans.PriceDataBean prodPrice =  
product.getCalculatedContractPrice();  
        %>  
        <tr>  
            <td width="10">&nbsp;   </td>  
            <td align="left" valign="top" width="280" class="categoryspace">
```

```

        <font class="product"><a
href="ProductDisplay?catalogId=<%=catalogId%>&storeId=<%=storeId%>&productId=<%=
=product.getProductID()%>&langId=<%=languageId%>&parent_category_rn=<%=parentCa
tegoryId%>">
        " hspace="5" width="50"
height="50" border="0" align="left">
        <%=product.getDescription().getShortDescription()%></a></font><br>
        <font class="price"><%=prodPrice%></font>
    </td>
</tr>
<%
    } catch (Exception e) {
%>
<tr>
    <td width="10">&nbsp;</td>
    <td align="left" valign="top" width="280" class="categoryspace">
        <font class="product"><a
href="ProductDisplay?catalogId=<%=catalogId%>&storeId=<%=storeId%>&productId=<%=
=product.getProductID()%>&langId=<%=languageId%>&parent_category_rn=<%=parentCa
tegoryId%>">
        " hspace="5" width="50"
height="50" border="0" align="left">
        <%=product.getDescription().getShortDescription()%></a></font><br>
        <font class="price">#no price#</font>
    </td>
</tr>
<%
    }
}

```

ProductDisplay.jsp

The ProductDisplay page also shows the price on the product bean, so you will need to modify this page as well. Even under normal circumstances its a good idea to handle exceptions in your JSP files. There is always the risk that a product is missing a description or a price and if you do not handle exceptions the page will not display correctly or will not display at all.

Remember that it is necessary for any possible exception to be thrown in your try-catch before any HTML is printed. See Example 11-18.

Example 11-18 ProductDisplay.jsp

```

<%
try {
    com.ibm.commerce.price.beans.PriceDataBean prodPrice =
product.getCalculatedContractPrice();

```

```
%>
  <font class="price"><%=prodPrice%></font><br><br>
<%
} catch (Exception e) {
    // do nothing.
}
%>
```

11.8 Final considerations

What we have done in this chapter is only the top of the iceberg when it comes to the possibilities that lie in the messaging subsystem. The code made in this chapter only supports import of products, items and category relations. But it could very easily be extended to support import of prices. Since the current WebSphere Commerce price import facility only supports update of existing prices it makes it relevant to support import of new prices along with the items.

11.8.1 The markfordelete attribute

Only use the markfordelete attribute if you really want to delete the entity from the database. In most cases it should be sufficient to just set published=0 on its description. However when using conventional massloading of the data you won't be able to specify that you wish to remove a relation to a category. In that case the only solution would be to delete the product and create it again.

11.8.2 Inventory

When we are creating products the WebSphere Commerce command automatically creates an inventory entry for the products and items with a quantity of 9999 units. You will be able to use the built-in message called Update_WCS_ProductInventory. This message has a few disadvantages. It does not support creation of new inventory entries, and you cannot specify the fulfillment center by its name instead of the internal WebSphere Commerce generated id.

11.8.3 Performance

We have not done performance tests on the MQProductCreate command. But all things being equal when compared to the conventional massloader the MQProductCreate command will affect both the application server and the database whereas the massloader will only affect the backend.

11.8.4 SKUs and attributes

If you create a new or delete an attribute on a product, then all of the items associated with the product will be deleted. With MQProductCreate it does not pose a direct problem since the items are recreated right after they have been deleted - but if you have created any prices manually they will be deleted.

You should be very careful when specifying attributes for your products. They should be created only the first time the product is loaded and then remain fixed.

We have not implemented for example auxdescription in the MQProductDescriptionCmd, so if you want to use the aux fields then you should perform the necessary changes to the command.

11.8.5 Transactions

The imports done with the MQProductCreate are performed in the same transactions which means that should something go wrong somewhere in the process it will roll everything back. With this in mind it could be a good idea, even though the MQProductCreate supports more than one product in the same file, to send one message per product. This means that should something go wrong with one product then all the rest will still be loaded.



B2B features

In this chapter we customize the ToolTech sample store to enable some business-to-business scenarios, including:

- ▶ Allowing the customer to select a contract for the entire shopping experience, as opposed to selecting a contract while adding items to the shopping cart.
- ▶ Role-based welcome or home page
- ▶ Request for order approval if the order exceeds a certain value

This chapter also covers the message extensions features available in WebSphere Commerce Business Edition V5.4.

12.1 B2B features in WebSphere Commerce Business Edition V5.4

WebSphere Commerce Business Edition is designed to provide additional business-to-business features need to implemeneted a sophisticated e-commerce site. This section details some of the main B2B features that are available.

12.1.1 Access control

WebSphere Commerce Business Edition, has an improved access control system that provides a hierarchical structure and a role based control. The access control policies are fine grained so they can provide control over the instance. Implementation of access control is externalized so if case you want to customize access control, you will not have to make changes to the WebSphere Commercecode.

Member system

You can create profiles for customers, create member groups, associate customers to member groups, and specify access control policies for member groups. You can specify discounts for organizations. The structure of organizations is very hierarchical, so that you can have approvals at the appropriate levels, and you can also inherit properties from ancestors.

Order system

WebSphere Commerce Business Edition allows customer to process orders at the commerce site by providing purchase order numbers. Customized invoices can be sent to different customer/buyer organizations in a commerce site.

Pricing is based on contracts and every customer organization has a contract associated with it. The price at which the product will be supplied to customers belonging to a buyer organization will depend on the contract. Payment methods and shipping methods can also be controlled by contracts.

Customer's may create requisition lists that can either be private or public. By creating requisition list's customers can create or place orders without browsing through the catalog. You can also create recurring orders, specifying an order frequency and payment method.

Orders can be split to two different order's if the required inventory is not available. The split order will be fulfilled once the inventory is available.

You can configure the commerce server to send e-mail notification when an order status changes - this feature can be extended to send e-mail when a order is awaiting approval.

Inventory system

The inventory system in the commerce server provides real times inventory management solutions. It provides an interface to persist inventory information received from vendors and the inventory received from the customers. It can adjust inventory quantity, and ship and receive inventory.

Request for quote(RFQ)

WebSphere Commerce Business Edition allows buyer organizations to raise a RFQ for large quantity purchases of specific items. Once a RFQ is posted the seller organization can send a response by creating a RFQ response. An order request can be directly created from a RFQ response. The process flow for RFQ handling can be modified by changing XML files.

Search system

Basic and advanced search catalog features are available in WebSphere Commerce Business Edition. The product advisor feature helps customers to identify products by a drill down process.

12.2 Role-based display

Customers in WebSphere Commerce Business Edition, are assigned role's and perform actions based on their assigned roles. The list of actions that customers can perform on a given resource is controlled by access control policies.

Roles are defined with respect to organizations. When a role is assigned to a member this member plays specific roles in a specific organization. A member may not be assigned any roles which are not entitled for the organization. For example, a member cannot have a buyer approver role, if the members's organization does not have this role. The list of roles that are available in the commerce server are stored in the ROLE table. You can use the available roles and assign roles to customers as well as create new role's and assign them to members. To create new role, use the Administration Console. The associations of roles and members are maintained in the MBRROLE table.

The steps to assign roles to customers are discussed in Chapter 5, "Creating a store" on page 131.

In our role based sample, we create a welcome page, that displays a link to Order Approval and to RFQ, if the logged in customer has a role of a buyer approver or buyer administrator in the customer organization. The sample store that we will be using to showcase this feature is ToolTech. For more details about this sample store refer to the online help for ToolTech

The modified home page will contain a link to approve orders and a link to create RFQs if the logged in customer is a buyer approver or a buyer administrator. These links will not display if the user is just a buyer and does not have any administration role in the buyer organization.

This feature is implemented in the ToolTech sample store by using the following JSPs:

- ▶ LogonDisplay.jsp
- ▶ CatalogMainDisplay.jsp

To retrieve information about registered users from the commerce server, use the `UserRegistrationDataBean`. This bean can be initialized by passing an instance of the bean to the `DataBeanManager` along with the request object. The `DataBeanManager` takes care of setting initialization parameters and invoking the `populate` method of the bean instance.

The `UserRegistrationDataBean` has method `getRoles()` which returns an array of type `Integer` containing the roles this customer performs with respect to the parent organization.

On our modified home page page, we get the roles of the logged in user and display the appropriate content based on the roles. Example 12-1 shows code from `CatalogMainDisplay.jsp`

Example 12-1 Sample to retrieve user roles

```
UserRegistrationDataBean bnRegUser = new UserRegistrationDataBean();
com.ibm.commerce.beans.DataBeanManager.activate(bnRegUser, request);
Integer [] userRoles = bnRegUser.getRoles();

//Get User's Role and determine whether to display the Buyer Approver link
boolean bBuyerApprover = false;
boolean bBuyerBuySide = false;
boolean bBuyerAdmin = false;
for (int i=0; i < userRoles.length; i++) {
    RoleDataBean dbRole = new RoleDataBean();

    //Using setInitKey_RoleId here instead of setRokeId because the databean
    //won't activate properly. setInitKey_RoleId is from RoleAccessBean. To
    comply
```

```

//with models and access control, we do not use RoleAccessBean.
dbRole.setInitKey_RoleId (userRoles[i].toString());
DataBeanManager.activate(dbRole, request);
if (dbRole.getRoleId().equals ("-22")) {
    System.out.println("bBuyerApprover");
    bBuyerApprover = true;
} else if (dbRole.getRoleId().equals ("-24")) {
    bBuyerBuySide = true;
    System.out.println("bBuyerBuySide");
} else if (dbRole.getRoleId().equals ("-21")) {
    bBuyerAdmin = true;
    System.out.println("bBuyerAdmin");
}
}
}

```

In the code shown in Example 12-1 notice that on retrieval of user roles, we are checking if the roles match any of the administrative privileges for the organization. If the user has administrative authority, we set on a flag. Based in the status of the flag the content displayed by the JSP is varied as shown in Example 12-2

Example 12-2 Sample to dynamically change content based on role

```

<P>
    <% if (bBuyerBuySide && (bBuyerApprover || bBuyerAdmin)) { %>
    <hr width="580" noshade align="left">
    <P>
    <b><font color=#ffffff><A CLASS="catalog" HREF="javascript:
ApprovalToolLink();"><%= tooltechtext.getString ("Home_Link1") %> </A></font>
</b>
    <% } %>
    <P>

```

Figure 12-1 shows the ToolTech home page

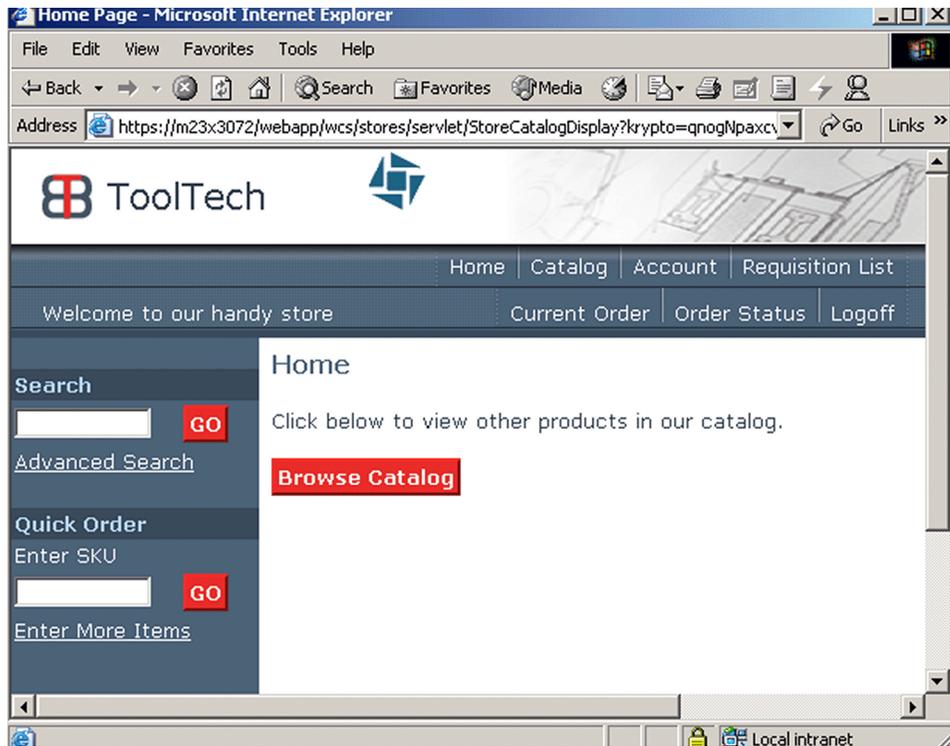


Figure 12-1 ToolTech home page

The content of the home page shown in Figure 12-1 is dynamically changed based on the user role. If the user has the appropriate roles the home page will display a link to approve pending orders for the customer organization the buyer belongs to and a link to create new RFQs. Figure 12-2 is a screen sample of home page for a buyer with approver and admin roles. Figure 12-1 is the home page for a buyer who has no administrative privileges in the buyer organization.

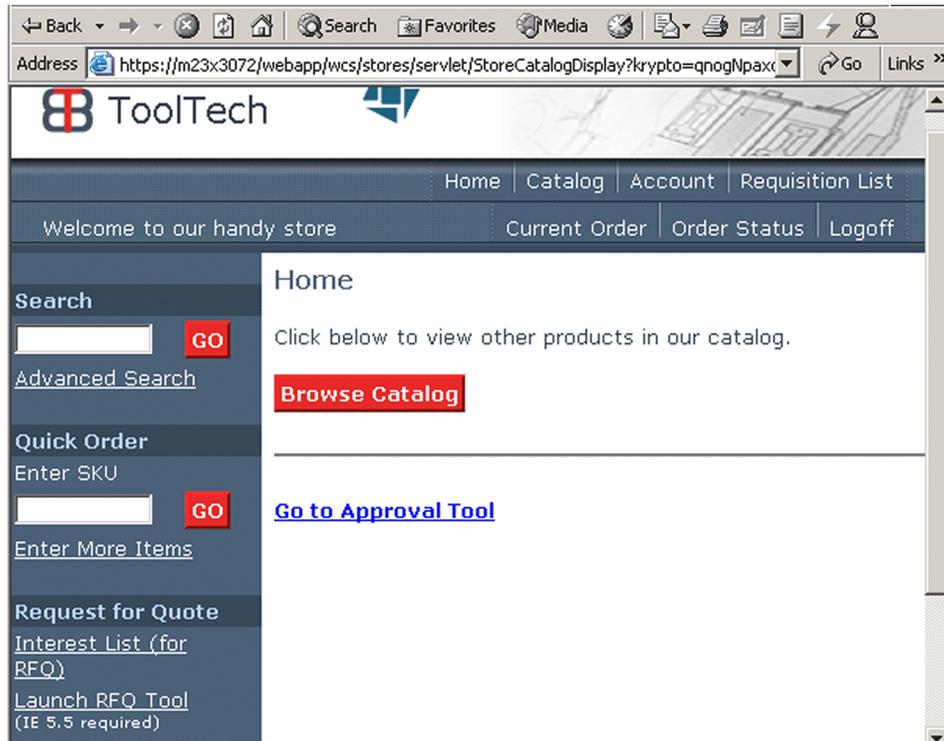


Figure 12-2 ToolTech home page for approvers.

Using similar techniques you can also dynamically change the content of the view based on business profile of users. For example if the buyer belongs to a customer organization that is only interested in subsets of the store catalog, you can customize the view accordingly. In addition to this you can also create custom roles and change the display content accordingly.

12.3 Order approval

An order may be placed for approval before processing based on terms and conditions defined by contracts. When a product is added to the shopping cart WebSphere Commerce Business Edition checks whether the terms and conditions defined in the contract require order approval for products purchased under this contract. If approval is required the order is put into approval pending state otherwise the order is placed for further processing. Only after an approver from the buyer organization approves an order that is in approval pending state can the order be placed for further processing.

Details about how to create contracts have already been discussed in chapter Chapter 5, “Creating a store” on page 131. To set up a contract that requires order approval based on the order amount you can specify this while creating a contract. You use the order approval section of the contract creation process as shown in Figure 12-3. While specifying a minimum amount for order approval make sure you mention it for all the currency formats the store is going to support.

The screenshot displays the 'Contract Order Approval' configuration page in the WebSphere Commerce administration console. The page title is 'Contract Order Approval'. A checkbox labeled 'Approval required' is checked. Below it, there is a field for 'Minimum amount (required)' with the value '250' and a dropdown menu for 'USD'. The left sidebar shows a navigation menu with 'Order Approval' highlighted. The top navigation bar includes 'Store', 'Sales', 'Marketing', 'Products', 'Logistics', and 'Help'. The top right corner shows the breadcrumb 'Logout > Home > Accounts > Contracts > Change Contract'.

Figure 12-3 Terms and conditions of contract

To modify an existing contract, make a new version of the contract, make the necessary changes and then publish the contract. The existing contract is modified and is set to active state.

To showcase this feature of contracts in WebSphere Commerce Business Edition, we use ToolTech as our sample store model. We have created two customers for buyer organization b, buyer1 and buyer2. buyer1 is an approver for buyer organization b, and buyer 2 does not have any roles assigned. For the steps to create users and assign role refer to Chapter 5, “Creating a store” on page 131. Figure 12-4 shows a view of these users in the WebSphere Commerce administration console.

Users

Page Number

5 items

« [First](#) | 1 of 1 | [Last](#) »

<input type="checkbox"/>	Logon ID	Last Name	First Name	Organization	Role	Find
<input type="checkbox"/>	seller1	Seller1		Seller Organization A	Seller	New
<input type="checkbox"/>	buyer1	Buyer1	ToolTech	Buyer Organization B	Buyer (buy-side), Buyer Approver, Buyer Administrator	Change
<input type="checkbox"/>	buyer2	Buyer2	ToolTech	Buyer Organization B		Roles
<input type="checkbox"/>	buyer3	Buyer3	ToolTech	Buyer Organization C	Buyer (buy-side), Buyer Approver, Buyer Administrator	Member Groups
<input type="checkbox"/>	wcsadmin	wcsadmin		Root Organization	Site Administrator	

Figure 12-4 Administration Console - user management)

When buyer2 places an order in the ToolTech store, with order value exceeding \$250, and using the contract ToolTech contract 6789 [10105] as shown in Figure 12-5 on page 334, the order is placed in an awaiting approval state. The contract that is used here is not the store default contract which is created while publishing the store.

Select a contract for this order in the table and click on the Add to Order button to add the contract to your order.

	Contract	Price
<input type="radio"/>	ToolTech Contract number 3456 [10005]	\$431.25
<input type="radio"/>	ToolTech Contract number 4567 [10006]	\$287.50
<input type="radio"/>	ToolTech Contract 1234 [10104]	\$517.50
<input checked="" type="radio"/>	ToolTech Contract 6789 [10105]	\$546.25

Add to Order

Figure 12-5 Contracts In ToolTech store

After placing an order, the order status is set to pending approval state. An order confirmation will not be displayed, but a status message will be displayed as shown in Figure 12-6. This status message will not be displayed if there are no approvers associated to the buyer organization..

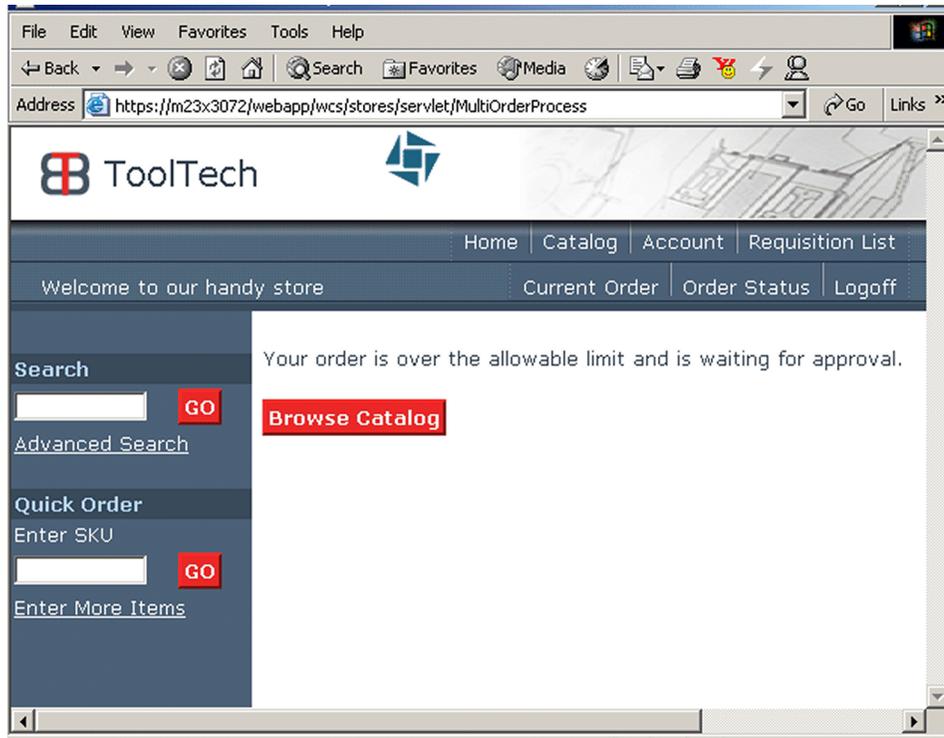


Figure 12-6 Order confirm page shown approval pending status

To check order status select the **Order Status** link in the toolbar of the ToolTech store. Figure 12-7 will be displayed.

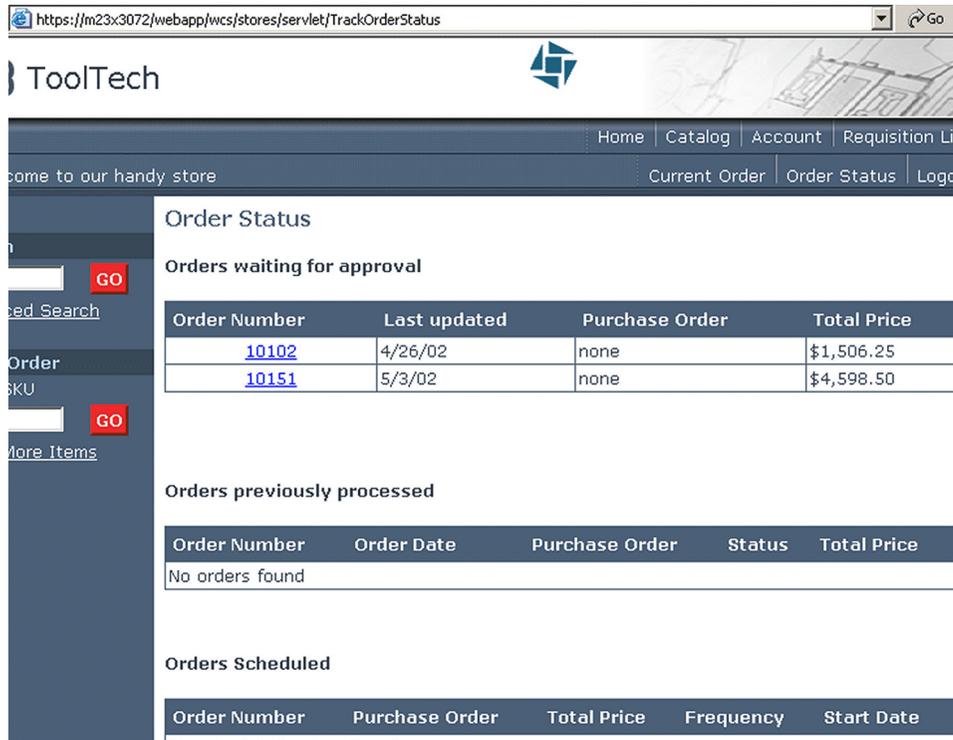


Figure 12-7 OrderStatus JSP

The order approval page is built based on the WebSphere Commerce tools framework. For more information on customizing and building tools using the framework refer to the *WebSphere Commerce Tools Framework Programmer's Guide*.

You can use the existing approval JSPs as is, or customize the JSPs to meet your business needs. If you have published the ToolTech sample store, can find the approval JSPs in
 <WAS_DIR>\installedApps\WC_Enterprise_App_demo.ear\wcstores.war\tools

Note: Make sure you have assigned appropriate roles to organizations and to users. If you have not assigned the correct roles you may not be able to test the approval feature. By default, ToolTech does not assign any roles to users or organizations. For instructions on how to assign roles to members refer to Chapter 5, "Creating a store" on page 131

When you log into the ToolTech store as a buyer organization approver you will see a link to the order approval JSP. Using the organization order approval page as shown in Figure 12-8, you can update organizational details, assign roles to users, and approve or reject orders.

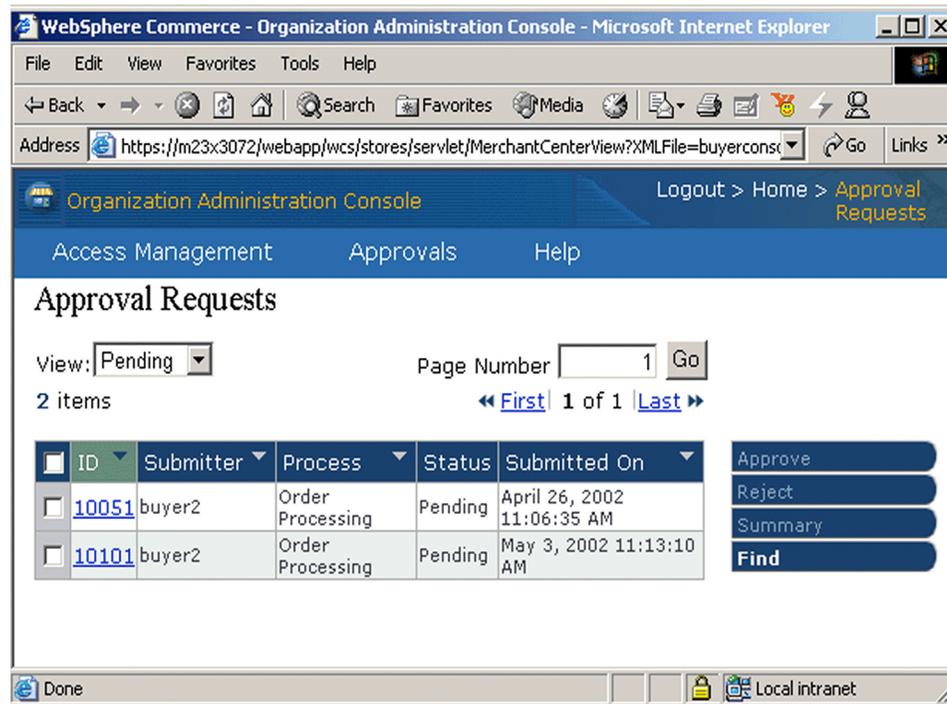


Figure 12-8 OrderApproval Page

Tip: Make sure you have created a buyer approver or buyer administrator, otherwise you may not be able to place an order for approval.

12.4 Contracts and trading agreements

In WebSphere Commerce Business Edition the pricing of product items is based on contracts; the same item can be sold at different prices by having multiply contracts. A contract cannot exist without having a business account. A business account is associated with customer organizations. The procedure to create business accounts and contracts is discussed in Chapter 5, “Creating a store” on page 131

In the shopping flow that the default store models use, the list of eligible contracts is shown in the item display page. In this section we customize the sample store to modify the basic shopping flow so that the customer will be able to view the list of contracts that she is eligible to use when the store home page is displayed. The customer can choose a contract which will be the default contract for the entire shopping experience.

The use case of this scenario is as follows:

Preconditions:

Registered users log on to the system, by supplying their authentication information.

Basic flow:

- ▶ The system displays the Logon page.
- ▶ The customer selects a shopping language and enters their user ID and password in the User ID and password fields.
- ▶ The customer clicks **Submit** and the information is submitted.
- ▶ After log on is finished the customer is transferred to the home page of the store. The home page displays the list of eligible contracts.
- ▶ The user can select a particular contract, and the selected contract will be set as the default contract during the entire shopping experience.

Alternate flow:

- ▶ The customer forgets their password and would like to reset their password.
- ▶ The customer selects the **Forgot Your Password?** link.
- ▶ The system displays the forgot password page.
- ▶ The customer enters their User ID in the appropriate field.
- ▶ The customer clicks **Send My Password** and the information is submitted.
- ▶ The system sends the password to the customer's e-mail address.
- ▶ A confirmation message is displayed.

- ▶ The customer can transfer to the Log on page after they have received their password.

Exception flow:

If any exception occurs the user is redirected to the generic error JSP and the appropriate error message is displayed.

Contracts are associated with a business account which links to customer organizations known to the commerce server. Users of a customer organization will be able to use contracts associated to the business account of their customer organization. The contracts can specify a list of items that a covered under the contract, payment methods, shipping modes and other pricing information. To create a contract please refer to Chapter 5, “Creating a store” on page 131. To specify the list of items that can be purchased under a contract:

1. While creating a contract click **Selection Constraints**. You can either specify categories and items to be included in the contract or you can specify the categories and items to be excluded from the contract. This is shown in Figure 12-9

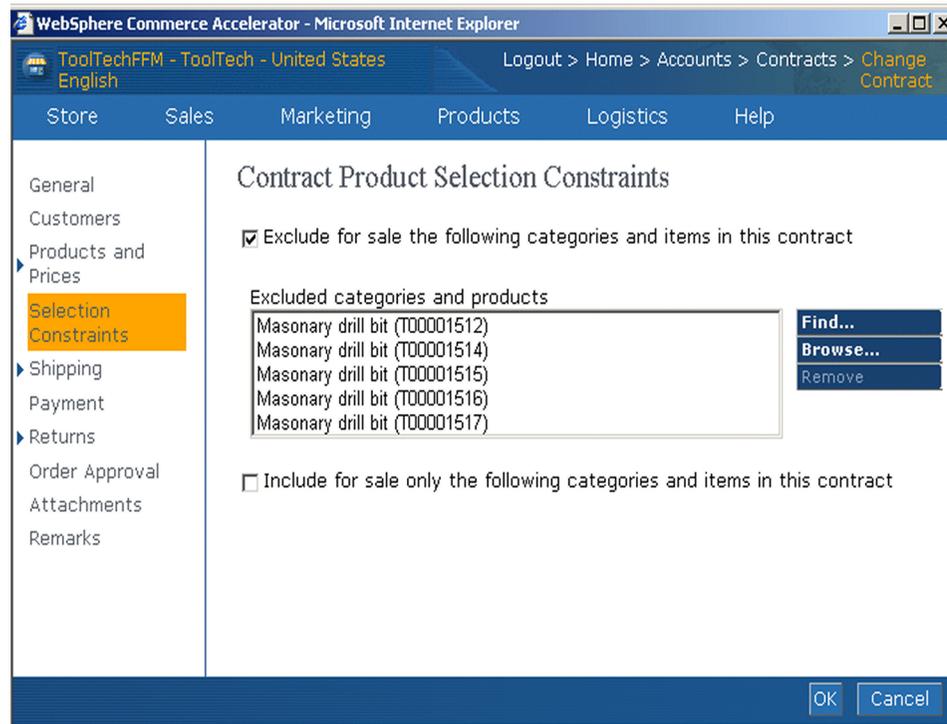


Figure 12-9 Selection constraints for contracts.

2. To select the list of categories or items you want to exclude or include in the contract you can either do a find for the particular category or item, or you can browse the store catalog and select the required items. In our example we select to exclude masonry drill bits from the contract. Figure 12-10 shows the browse functionality used to make our selections.
3. In the default contracts created with the ToolTech store, no restriction is applied, so you may browse through the entire catalog. To restrict some items we created a new contract. For details of creating a new contract refer to Chapter 5, “Creating a store” on page 131.

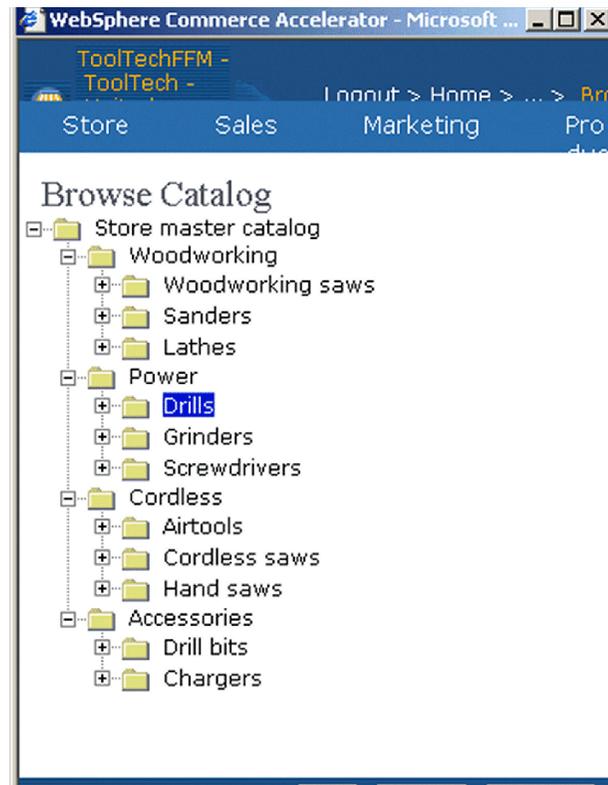


Figure 12-10 Select category or item

- ▶ In order to restrict displaying categories and items which cannot be purchased under a contract, you can either modify an existing contract or create a new contract as we did in our example. When the user logs on to the system and selects a contract to be the default contract for the entire shopping experience, only the categories or items that can be purchased under the contract will be displayed. As a consequence the user may not be

able to see all the categories and items listed in the catalog. The following sections describe how to implement this functionality.

You can extract information about the logged in users contract entitlements from the command context as shown in Example 12-3.

Example 12-3 Get eligible contracts from the command context.

```
String[] contractIds = null;
com.ibm.commerce.contract.objects.ContractAccessBean acBean = null;

try{
    contractIds = cmdcontext.getEligibleTradingAgreementIds();
    acBean = new com.ibm.commerce.contract.objects.ContractAccessBean();
    for(int i=0 ; i < contractIds.length ; i++)
    {
        acBean = new com.ibm.commerce.contract.objects.ContractAccessBean();
        acBean.setInitKey_referenceNumber(contractIds[i]);
        acBean.refreshCopyHelper();
    }
}
catch(Exception e)
{
    e.printStackTrace();
}
```

Using this technique we modified the ToolTech home page to display the list of eligible contracts to the logged in user. We use the same code show in Example 12-3 to extract information about eligible contracts. The user may select a contract from the list displayed and this contract set is as the default contract for the entire shopping experience. To make a contract as the default for session we use the `ContractSetInSession` command. Using this command you can set one or more contracts to be the default contracts for the session. As a result, in the item display page the price of items displayed will be only for the default contracts and not all eligible contracts.

Figure 12-11 is the modified home page of ToolTech store. A list of eligible contracts for the user who is currently logged in is displayed, and the user can select a particular contract and set it as the default contract. In our example we only set a single default contract, but the example can be extended to have multiple default contracts for the session. To make this change syou would have to pass multiple contract ids instead of only one id.

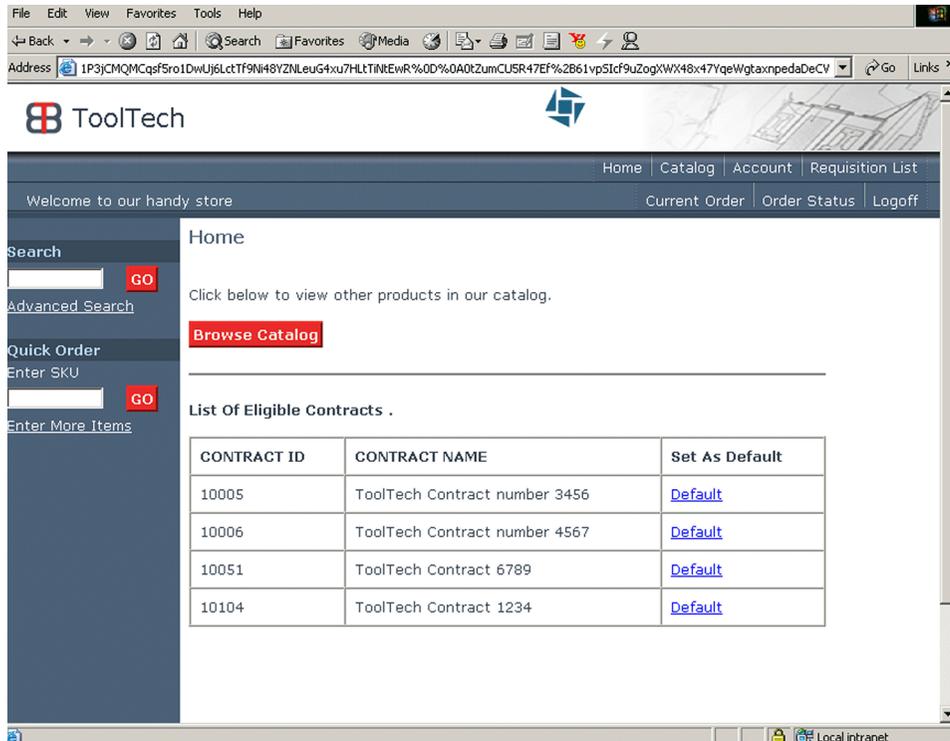


Figure 12-11 ToolTech home page with contracts list

After the user selects a contract to set as the default contract he user is redirected to a page where details of the contract are displayed. The user can change the default contract again by returning to the home page (using the Home button on the navigation bar), and selecting a different contract from the list.

The controller command that we used to set default contracts for the session is `ContractSetInSession`. This command can take one or many contract ids and set these contracts to be the default contracts for the session. Usage of this command is show in Example 12-4.

Example 12-4 ContractSetInSession command usage

This example allows logged in user to shop with contract ids 10001 & 10002.

`https://hostname/webapp/wcs/stores/servlet/ContractSetInSession?contractId=10001&contractId=10002`

This command sets the default contracts for user into the session. The contract ids specified in the command will be used if this command is successfully executed and the other contracts that the user is eligible to use will be ignored. On successful completion of this command the ContractListView is called. This view command by default point to <store_dir>/tools/contract/ContractList.jsp.

Based on your business needs you can modify the view command to point to a different JSP. To modify a view command you can either modify an existing view command to point to different JSP or you can create a version of this view command that is specific to your store. Example 12-5 on page 343 shows the SQL you can use for creating a new version of ContractListView command. Remember to change 10051 to the correct value for your STOREENT_ID.

Example 12-5 SQL to create view command used in this sample

```
INSERT INTO VIEWREG
(VIEWNAME,DEVICEFMT_ID,STOREENT_ID,INTERFACENAME,CLASSNAME,PROPERTIES,DESCRIPTI
ON,HTTPS,INTERNAL ) VALUES
('ContractListView',-1,10051,'com.ibm.commerce.command.ForwardViewCommand'
,'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=ContractListDis
play.jsp','View For Listing Eligible Contracts',1,1);
```

The list of contracts that the user is eligible to use is displayed in the home page using a modified version of CatalogMainDisplay.jsp. Refer to Example 12-3 on page 341 for sample code to retrieve eligible contracts using the command context. For our example we created a new JSP called ContractListDisplay.jsp and associated this ContractListView command for our store. This JSP will display details of the contract which the user has chosen to be the default contract.

Once user has set a contract to be the default contract, the categories and items that would be displayed to the user is controlled by the contract, that is the categories or items excluded in the contract will not be displayed to the user. The price displayed for the selected item will be the price as recommended in the chosen session default contract. Prices that are available to the customer with other eligible contracts are not displayed.

The contract we created for our example had selection criteria which excluded masonry drill bits. Figure 12-12 shows that these items are not being displayed to the logged in user.

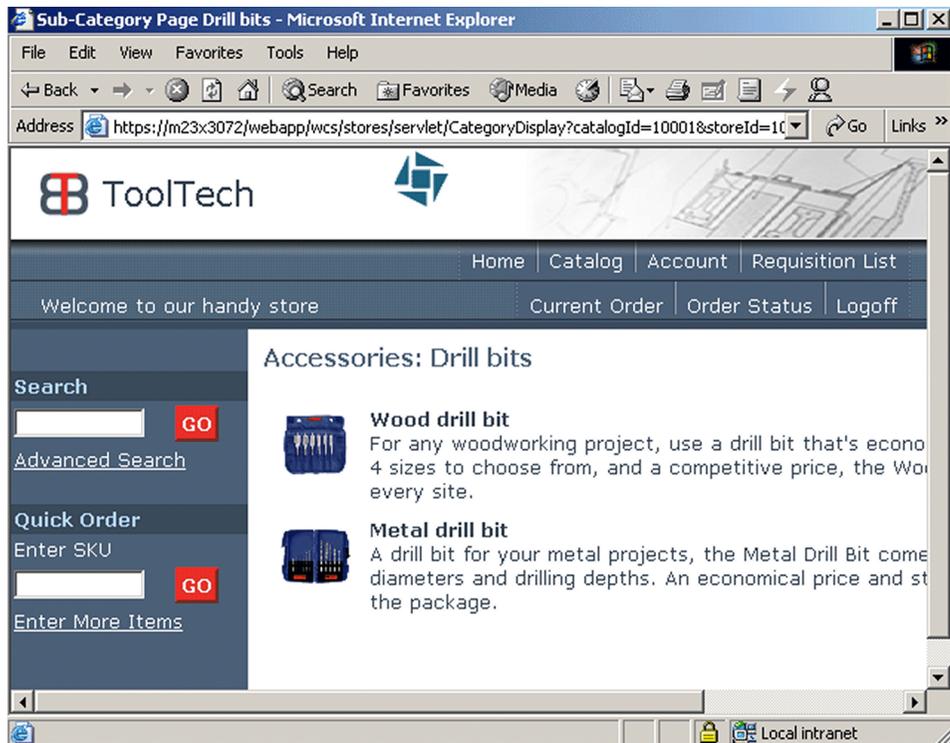


Figure 12-12 Masonry drill bits excluded.

Figure 12-13 shows these items being displayed to the logged in user when the user has chosen other contracts.

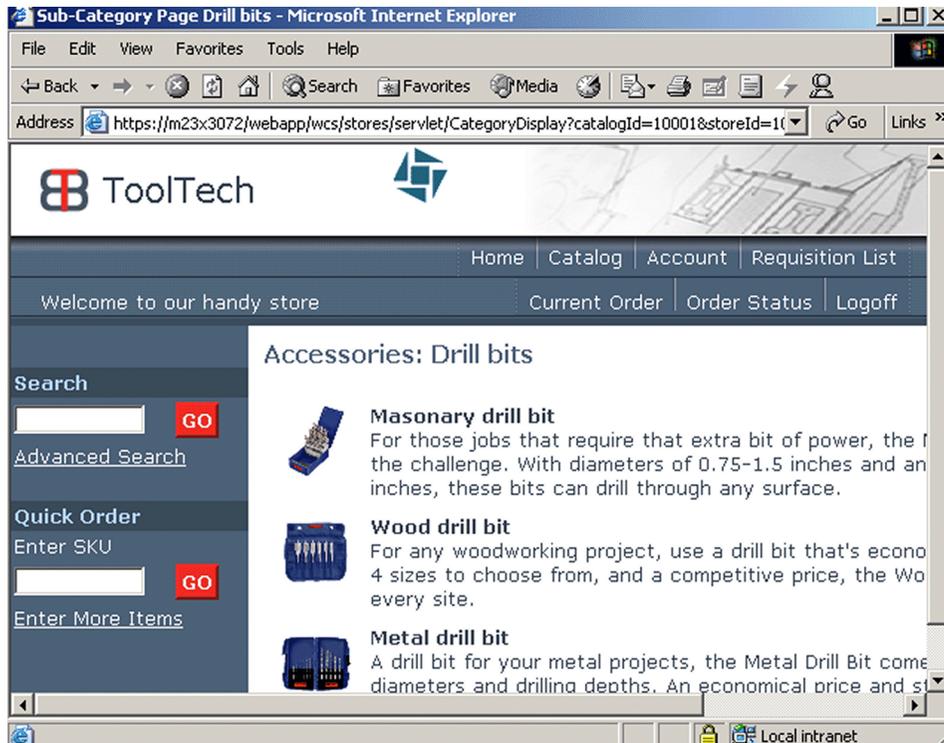


Figure 12-13 Masonry drill bits displayed for alternate contracts.

12.5 Message extensions

Message extensions is functionality built on top of WebSphere Commerce to provide extension points for integrating external buy-side systems. To achieve this, changes were made to WebSphere Commerce schema, business logic and the messaging system.

By using message extensions suppliers, can maintain an single catalog that can be presented to customers using a Web browser to shop and canalso be presented to customers using a procurement system such as Ariba Buyer. This helps you integrate WebSphere Commerce with the back end systems like supply chain management and order management. The aim is to reduce the effort and cost of in order processing and fulfillment.

Message extensions provides:

- ▶ DUNS number support for buyer and seller organizations
- ▶ UNSPSC based classification for categories and products.

- ▶ Contract based pricing for different buyer and seller organizations.
- ▶ Shopping cart can be sent to the buyer system for approval.
- ▶ Purchase order request can be received from an external system and processed in WebSphere Commerce
- ▶ Buyers can check their order status from the buy-side systems, and status responses can also be sent to buyer's systems.
- ▶ Buyers can check for product availability before placing an order

The end to end flow in a sell- side extension is as follows:

1. An customer from the buyer organization logs into the procurement system providing individual authentication information. A list of external catalogs from sell-side systems are displayed.
2. The front-end system (procurment system), validates the user, and whether he or she belongs to the organization. Once the user is validated, a request is constructed and sent to the supplier system, with authentication and organizational credentials.
3. The supplier system, which in this case is WebSphere Commerce, validates the supplied credentials. Once she is sucessfully authorized, the buyer is redirected to the URL which displays the suppliers catalog entries and products.
4. Buyers browses through the catalog and create a shopping cart, once the buyer submits the shopping cart, a response is sent back to buyer approver who has to confirm whether this purchase order request can be approved for further processing by the message extension system.
5. If the buyer approver edits the shopping cart the remote catalog punchout session starts again.
6. After buyer approver approves the order, the order approval is sent back to the message extension system, and propagated to WebSphere Commerce.

12.5.1 cXML overview

cXML allows parties in an e-commerce transaction such as buyers, suppliers, service providers, retailers, distributors, and so on, to communicate with each other using a standard language. To have a sucessful business-to-business solution it is helps to have an open standard protocol which can improve interoperability between systems. cXML is designed to meet this requirement of the e-commerce industry, and also has been designed to support business-to-business commerce.

Some of the cXML documents that commonly used by systems include:

- ▶ Catalog
- ▶ Punchout
- ▶ Purchase orders

Catalog cXMLs are files which describe the products offered by a supplier and the price at which they are offered. In order to have a standard classification of categories and products, suppliers may use UNSPSC based classification. These catalogs can be used by buyers connecting to the suppliers view of procurement applications.

A supplier can have a punchout site which can provide interactive catalogs on the Internet. The punchout sites can communicate with a procurement system by using cXML. When a buyer connects to a punchout site, a punchout session is initiated and the buyer can view the supplier's punchout site and shop. Once the buyer finishes creating a purchase order request, the order information is returned to the procurement system. This setup can also be chained, where a market place receives a request from the procurement system and propagates the request to supplier systems and the response is propagated back to the procurement system.

12.5.2 cXML in WebSphere Commerce Business Edition

WebSphere Commerce uses cXML messages to integrate with the Ariba procurement solution. The list of cXML messages being used includes:

- ▶ PunchOutSetupRequest
- ▶ OrderRequest
- ▶ PunchOutSetupResponse
- ▶ PurchaseOrderMessage
- ▶ OrderResponse

New controller and task commands have been introduced in WebSphere Commerce Business Edition V5.4 in order to interact with procurement systems and also to setup an punchout site. Table 12-1 relates the message extension controller commands to the cXML messages they will be operating on.

Table 12-1 cXML - command mapping

Input Message Name	Message Extensions Commands	Direction In-Bound/Out-Bound	Response Message Name
PunchOutSetupRequest	PunchOutSetup	IN	PunchOutSetupResponse
OrderRequest	BatchOrderRequest	IN	OrderResponse

Input Message Name	Message Extensions Commands	Direction In-Bound/Out-Bound	Response Message Name
PutOutSetupResponse	PunchOutSetup	OUT	
PurchaseOrderMessage	PrepareOrder	OUT	
OrderResponse	BatchOrderRequest	OUT	



Product entry and display

This chapter describes product entry and display features and customization. We detail product comparison features, and also describe how to create new products and product bundles. We also describe how to display dual currencies in a WebSphere Commerce site.

13.1 Product comparison

Product comparison allows user on an e-commerce site to compare data from a set of products. Each product is represented as either a row or a column in the table, and the values for the product's features are shown in the cells of the table.

To get the product comparison data, WebSphere Commerce Business Edition can use the included Product Advisor tool.

13.1.1 Product Advisor

The Product Advisor is a tool used to create an online product catalog that provides shoppers with different ways of finding the products they want. The alternative methods of finding products are called shopping metaphors. The Product Advisor can use the product comparison metaphor to compare similar products side by side with the same attributes like color or size.

To get product information there are two search methods available for use with the Product Advisor:

- ▶ Separate search space
- ▶ Base search space

The separate search space requires additional tables added to the database to reformat existing product data. This enables optimized parametric searches, which are focused on individual categories.

The base search space searches the WebSphere Commerce database.

To use these search methods, you have to populate some additional attribute metadata in the database.

The Product Advisor includes a command utility, named PAConfig, to help automate the process of creating either search space.

13.1.2 Creating base search space

The search space consists of additional database tables that contain information extracted from the following standard WebSphere Commerce tables:

- ▶ CATENTRY
- ▶ CATENTDESC
- ▶ CATGPENREL
- ▶ ATTRIBUTE
- ▶ ATTRVALUE
- ▶ LISTPRICE

For more informations about the tables refer to the WebSphere Commerce Version 5.4 Online Help.

Attribute definitions in the search space XML file

Product Advisor supports all the attributes in the ATTRIBUTE table. To add attributes to a search space using a XML file and the PAConfig utility, add the following information:

- ▶ The <columnName> is the value in the NAME column in the ATTRIBUTE table, for example Size for the English version.
- ▶ The <attrName> is the value from the NAME column for the language identified. This value is case sensitive. For example, Talla for the Spanish version.
- ▶ The Product Advisor also support attributes from other tables, such as CATENTRY and CATENTDESC. To add attributes to a search space with a XML file using the PAConfig utility add information such as that shown in Table 13-1.

Table 13-1 Base search space

Attribute	<columnName> value	<attrName> value
<i>Catentry_id</i>	CATENTRY_ID	CATENTRY_ID
<i>SKU (part number)</i>	PARTNUMBER	PARTNUMBER
Short description	SHORTDESCRIPTION	SHORTDESCRIPTION
Thumbnail	THUMBNAIL	THUMBNAIL
XML Detail	XMLDETAIL	XMLDETAIL
Price	LISTPRICE	LISTPRICE or PRICE
Availability	AVAILABLE	AVAILABLE

Note: Attributes shown in the *Italic* font in Table 13-1 are required for the search space.

Follow these steps to create a base search for the Product Advisor:

1. Choose a category for which you want to enable a Product Advisor search, in our example the category is the 10101.
2. Create an XML input file based on the sample XML file located in <WCS_DIR>\CommerceServer\samples\pa\xml
This sample XML file is called ss10003.xml for the category 10003.
3. Copy and rename the file, for example we created ss10101.xml.

4. Define the attributes for the base search space. In our example the category is 10001 from the WebFashion sample store. We use both languages provided by the WebFashion sample, that is English and Spanish. The attributes we defined are:

- Catentry_id
- part number
- short description
- thumbnail
- XML Detail
- Available
- Size

Attention: Check the attributes for the category. Ensure that you use the same attributes for all products or items in the category.

You can list all product attributes and their types for the category 10101 with the sql-statement :

```
select distinct(attribute_id),language_id,attrtype_id,name
from attribute
where
catentry_id in(select catentry_id from catgpenrel where
catgroup_id=10101)
order by attribute_id
```

5. Change the category id and add the attributes into the ss10101.xml as shown in the example 13-1:

Example 13-1 sample XML for search space

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalogBuilder SYSTEM "cbbatch.dtd">
<catalogBuilder>
  <delete_catalogBuilder CID="10101" />
  <category CID="10101">
    <attribute>
      <columnName>CATENTRY_ID</columnName>
      <length>8</length>
      <include>1</include>
      <type>com.ibm.commerce.pa.datatype.DsLong</type>
      <NLVdesc>
        <language>-1</language>
        <attrName>CATENTRY_ID</attrName>
        <description>Product Reference Number</description>
        <scale>0</scale>
        <precision>0</precision>
        <unitOfMeasure></unitOfMeasure>
      </NLVdesc>
    </attribute>
  </category>
</catalogBuilder>
```

```

    <NLVdesc>
      <language>-5</language>
      <attrName>CATENTRY_ID</attrName>
      <description>Product Reference Number</description>
      <scale>0</scale>
      <precision>0</precision>
      <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
  </attribute>
  <attribute>
    <columnName>PARTNUMBER</columnName>
    <length>64</length>
    <include>1</include>
    <type>com.ibm.commerce.pa.datatype.DsString</type>
    <NLVdesc>
      <language>-1</language>
      <attrName>PARTNUMBER</attrName>
      <description>Product Number/SKU</description>
      <scale>0</scale>
      <precision>0</precision>
      <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
    <NLVdesc>
      <language>-5</language>
      <attrName>PARTNUMBER</attrName>
      <description>Product Number/SKU</description>
      <scale>0</scale>
      <precision>0</precision>
      <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
  </attribute>
  <attribute>
    <columnName>SHORTDESCRIPTION</columnName>
    <length>254</length>
    <include>1</include>
    <type>com.ibm.commerce.pa.datatype.DsString</type>
    <NLVdesc>
      <language>-1</language>
      <attrName>SHORTDESCRIPTION</attrName>
      <description>Short Description</description>
      <scale>0</scale>
      <precision>0</precision>
      <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
    <NLVdesc>
      <language>-5</language>
      <attrName>SHORTDESCRIPTION</attrName>
      <description>Short Description</description>
      <scale>0</scale>

```

```

        <precision>0</precision>
        <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
</attribute>
<attribute>
    <columnName>THUMBNAIL</columnName>
    <length>254</length>
    <include>1</include>
    <type>com.ibm.commerce.pa.datatype.DsImage</type>
    <NLVdesc>
        <language>-1</language>
        <attrName>THUMBNAIL</attrName>
        <description>Thumbnail Image File</description>
        <scale>0</scale>
        <precision>0</precision>
        <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
    <NLVdesc>
        <language>-5</language>
        <attrName>THUMBNAIL</attrName>
        <description>Thumbnail Image File</description>
        <scale>0</scale>
        <precision>0</precision>
        <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
</attribute>
<attribute>
    <columnName>XMLDETAIL</columnName>
    <length>32700</length>
    <include>1</include>
    <type>com.ibm.commerce.pa.datatype.DsURLLink</type>
    <NLVdesc>
        <language>-1</language>
        <attrName>XMLDETAIL</attrName>
        <description>XML Detail</description>
        <scale>0</scale>
        <precision>0</precision>
        <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
    <NLVdesc>
        <language>-5</language>
        <attrName>XMLDETAIL</attrName>
        <description>XML Detail</description>
        <scale>0</scale>
        <precision>0</precision>
        <unitOfMeasure></unitOfMeasure>
    </NLVdesc>
</attribute>
<attribute>

```

```

<columnName>AVAILABLE</columnName>
<length>4</length>
<include>1</include>
<type>com.ibm.commerce.pa.datatype.DsInteger</type>
<NLVdesc>
  <language>-1</language>
  <attrName>AVAILABLE</attrName>
  <description>Availability</description>
  <scale>0</scale>
  <precision>0</precision>
  <unitOfMeasure></unitOfMeasure>
</NLVdesc>
<NLVdesc>
  <language>-5</language>
  <attrName>AVAILABLE</attrName>
  <description>Availability</description>
  <scale>0</scale>
  <precision>0</precision>
  <unitOfMeasure></unitOfMeasure>
</NLVdesc>
</attribute>
<attribute>
  <columnName>SIZE</columnName>
  <length>254</length>
  <include>1</include>
  <type>com.ibm.commerce.pa.datatype.DsString</type>
  <NLVdesc>
    <language>-1</language>
    <attrName>Size</attrName>
    <description>Size</description>
    <scale>0</scale>
    <precision>0</precision>
    <unitOfMeasure></unitOfMeasure>
  </NLVdesc>
  <NLVdesc>
    <language>-5</language>
    <attrName>Talla</attrName>
    <description>Talla</description>
    <scale>0</scale>
    <precision>0</precision>
    <unitOfMeasure></unitOfMeasure>
  </NLVdesc>
</attribute>
</category>
</catalogBuilder>

```

6. Be sure to back up the WebSphere Commerce Server database.
7. Save the file and copy it to the path <WCS_DIR>\CommerceServer\bin

8. Localize the configuration file for your instance, for example this will be demo.xml if you have created an instance using default values. The configuration files will be found in
 <WCS_DIR>\CommerceServer\instances\instance_name\xml\instance_name.xml
9. Copy the configuration file of WebSphere Commerce instance into
 <WCS_DIR>\CommerceServer\bin
10. From a Windows command prompt, enter :
 cd <WCS_DIR>\CommerceServer\bin
11. Run PAConfig by entering the command:
 paconfig XML_file configuration_file
 See Example 13-2.

Example 13-2 paconfig for the example XML file

```
paconfig ss10101.xml demo.xml
```

12. Note that the message telling you that the import has completed does not indicate success. Look at either the paconfig.log file, or the command line output for error messages. The paconfig.log file can be found in
 <WCS_DIR>\CommerceServer\logs. See Example 13-3

Example 13-3 paconfig.log of the example ss10101.xml

```
5/10/02 3:49:32 PM 1.1.16
5/10/02 3:49:32 PM buildDOMTree >>>>
5/10/02 3:49:32 PM
5/10/02 3:49:32 PM
5/10/02 3:49:32 PM
5/10/02 3:49:32 PM ProductFamily for Category 10101
5/10/02 3:49:32 PM
5/10/02 3:49:32 PM Checking if the search space already exists for category
10101
5/10/02 3:49:32 PM the SQL is SELECT * FROM ICROOTCAT WHERE ROOTCATEGORYID =
10101
5/10/02 3:49:32 PM Passed check.
5/10/02 3:49:32 PM
5/10/02 3:49:32 PM
5/10/02 3:49:32 PM columnName = CATENTRY_ID
5/10/02 3:49:32 PM length = 8
5/10/02 3:49:32 PM include = 1
5/10/02 3:49:32 PM type = com.ibm.commerce.pa.datatype.DsLong
5/10/02 3:49:32 PM adding language -1
5/10/02 3:49:32 PM language = -1
5/10/02 3:49:32 PM attrName = CATENTRY_ID
5/10/02 3:49:32 PM description = Product Reference Number
```

```

5/10/02 3:49:32 PM scale = 0
5/10/02 3:49:32 PM precision = 0
5/10/02 3:49:32 PM
5/10/02 3:49:32 PM adding language -5
5/10/02 3:49:32 PM language = -5
5/10/02 3:49:32 PM attrName = CATENTRY_ID
5/10/02 3:49:32 PM description = Product Reference Number
5/10/02 3:49:33 PM scale = 0
5/10/02 3:49:33 PM precision = 0
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLFEAT insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert CATENTRY_ID *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert CATENTRY_ID *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM columnName = PARTNUMBER
5/10/02 3:49:33 PM length = 64
5/10/02 3:49:33 PM include = 1
5/10/02 3:49:33 PM type = com.ibm.commerce.pa.datatype.DsString
5/10/02 3:49:33 PM language = -1
5/10/02 3:49:33 PM attrName = PARTNUMBER
5/10/02 3:49:33 PM description = Product Number/SKU
5/10/02 3:49:33 PM scale = 0
5/10/02 3:49:33 PM precision = 0
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM language = -5
5/10/02 3:49:33 PM attrName = PARTNUMBER
5/10/02 3:49:33 PM description = Product Number/SKU
5/10/02 3:49:33 PM scale = 0
5/10/02 3:49:33 PM precision = 0
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLFEAT insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert PARTNUMBER *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM

```

```

5/10/02 3:49:33 PM ***** ICEXPLDESC insert PARTNUMBER *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM columnName = SHORTDESCRIPTION
5/10/02 3:49:33 PM length = 254
5/10/02 3:49:33 PM include = 1
5/10/02 3:49:33 PM type = com.ibm.commerce.pa.datatype.DsString
5/10/02 3:49:33 PM language = -1
5/10/02 3:49:33 PM attrName = SHORTDESCRIPTION
5/10/02 3:49:33 PM description = Short Description
5/10/02 3:49:33 PM scale = 0
5/10/02 3:49:33 PM precision = 0
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM language = -5
5/10/02 3:49:33 PM attrName = SHORTDESCRIPTION
5/10/02 3:49:33 PM description = Short Description
5/10/02 3:49:33 PM scale = 0
5/10/02 3:49:33 PM precision = 0
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLFEAT insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert SHORTDESCRIPTION
*****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:33 PM
5/10/02 3:49:33 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert SHORTDESCRIPTION
*****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM columnName = THUMBNAIL
5/10/02 3:49:34 PM length = 254
5/10/02 3:49:34 PM include = 1
5/10/02 3:49:34 PM type = com.ibm.commerce.pa.datatype.DsImage
5/10/02 3:49:34 PM language = -1
5/10/02 3:49:34 PM attrName = THUMBNAIL
5/10/02 3:49:34 PM description = Thumbnail Image File
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM language = -5
5/10/02 3:49:34 PM attrName = THUMBNAIL

```

```

5/10/02 3:49:34 PM description = Thumbnail Image File
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLFEAT insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert THUMBNAIL *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert THUMBNAIL *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM columnName = XMLDETAIL
5/10/02 3:49:34 PM length = 32700
5/10/02 3:49:34 PM include = 1
5/10/02 3:49:34 PM type = com.ibm.commerce.pa.datatype.DsURLLink
5/10/02 3:49:34 PM language = -1
5/10/02 3:49:34 PM attrName = XMLDETAIL
5/10/02 3:49:34 PM description = XML Detail
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM language = -5
5/10/02 3:49:34 PM attrName = XMLDETAIL
5/10/02 3:49:34 PM description = XML Detail
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLFEAT insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert XMLDETAIL *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert XMLDETAIL *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM columnName = AVAILABLE

```

```

5/10/02 3:49:34 PM length = 4
5/10/02 3:49:34 PM include = 1
5/10/02 3:49:34 PM type = com.ibm.commerce.pa.datatype.DsInteger
5/10/02 3:49:34 PM language = -1
5/10/02 3:49:34 PM attrName = AVAILABLE
5/10/02 3:49:34 PM description = Availability
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM language = -5
5/10/02 3:49:34 PM attrName = AVAILABLE
5/10/02 3:49:34 PM description = Availability
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLFEAT insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert AVAILABLE *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert AVAILABLE *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM columnName = SIZE
5/10/02 3:49:34 PM length = 254
5/10/02 3:49:34 PM include = 1
5/10/02 3:49:34 PM type = com.ibm.commerce.pa.datatype.DsString
5/10/02 3:49:34 PM language = -1
5/10/02 3:49:34 PM attrName = Size
5/10/02 3:49:34 PM description = Size
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:34 PM language = -5
5/10/02 3:49:34 PM attrName = Talla
5/10/02 3:49:34 PM description = Talla
5/10/02 3:49:34 PM scale = 0
5/10/02 3:49:34 PM precision = 0
5/10/02 3:49:34 PM
5/10/02 3:49:35 PM ***** ICEXPLFEAT insert successful *****
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM

```

```

5/10/02 3:49:35 PM ***** ICEXPLDESC insert Size *****
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM ***** ICEXPLDESC insert Talla *****
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM ***** ICEXPLDESC insert successful *****
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM ***** ICROOTCAT *****
5/10/02 3:49:35 PM Inserting into ICROOTCAT table.
5/10/02 3:49:35 PM Inserting into ICROOTCAT table successful.
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM
5/10/02 3:49:35 PM ***** Finished for category 10101 *****

```

13. Check the tables ICEXPLFEAT, ICEXPLDESC, ICROOTCAT in the WebSphere Commercedatabase.

13.1.3 Preparing a product comparison metaphor

Once you create the required search spaces, you have to create a XML file for the product comparison metaphor and then import the XML file into the WebSphere Commerce database using the PABatch command utility .

Before creating an XML input file, you must determine the following for the catalog data:

- ▶ The store ID, for example 10101
- ▶ The category ID, for example 10101
- ▶ The features that correspond to the database column names, for example. Size

Product Advisor provides a sample XML input file for use with the product advisor command utility. The sample file metaphor.xml, which includes a product comparison metaphor is located in

<WCS_DIR>\WebSphere\CommerceServer\samples\pa\xml

To create a new XML input file, do the following:

1. Copy the metaphor.xml file into the directory
<WCS_DIR>\CommerceServer\bin
2. Rename the file. In our example we used m10101.xml.
3. Change the store id (10101), and the category id (10101) as shown in Example 13-4.

```
<store SID="10101">
<category ID="10101">
```

4. To build a product comparison metaphor within a category you have to add the following specific tags. These are also shown in Example 13-4.

```
<productComparer>
</productComparer>
```

5. Insert a template element between the metaphor begin and end tags to specify the JSP page that the metaphor will be based on:

```
<template>/webapp/wcs/stores/servlet/pc51.jsp</template>
```

6. Add features to the sample file:

```
<feature>
<columnName>Size</columnName>
<order>1</order>
<display>1</display>
<sort>1</sort>
</feature>
```

7. The complete sample code added to the m10101.xml file for the product comparison metaphor is shown in Example 13-4.

Example 13-4 sample metaphor XML input file - m10101.xml

```
<?xml version="1.0"?>
<!DOCTYPE builder SYSTEM "pabatch.dtd" >
<builder>
  <store SID="10101">
    <category ID="10101">
      <productComparer>
        <template>/webapp/wcs/stores/servlet/pc51.jsp</template>
        <feature>
          <columnName>Size</columnName>
          <order>1</order>
          <display>1</display>
          <sort>1</sort>
        </feature>
        <feature>
          <columnName>PARTNUMBER</columnName>
          <order>8</order>
          <display>2</display>
          <sort>1</sort>
        </feature>
        <feature>
          <columnName>THUMPNAIL</columnName>
          <order>9</order>
          <display>1</display>
          <sort>1</sort>
```

```

        </feature>
        <feature>
            <columnName>SHORTDESCRIPTION</columnName>
            <order>10</order>
            <display>1</display>
            <sort>1</sort>
        </feature>
    </productComparer>
</category>
</store>
</builder>

```

8. From a Windows command prompt enter the command:

```
cd <WCS_DIR>\CommerceServer\bin
```

9. Enter the command:

```
PABatchXML XML_file configuration_file
```

See Example 13-5

Example 13-5 PABatchXML for the example XML file

```
PABatchXML m10101.xml demo.xml
```

10. After executing the PABatch command you should get message similar to those shown in Example 13-6

Example 13-6 output in command window for PABatchXML

```

C:\Program Files\WebSphere\CommerceServer\bin>PABatchXML m10101.xml demo.xml

C:\Program
Files\WebSphere\CommerceServer\bin>C:\WebSphere\AppServer\java\bin\ja
va -classpath
/"C:\PROGRA~1\WEBSPH~1\COMMER~1\lib;C:\WebSphere\AppServer/classes
;C:\WebSphere\AppServer\java\jre\lib\ext;C:\PROGRA~1\WEBSPH~1\COMMER~1\lib\ibmj
c
efw.jar;C:\PROGRA~1\WEBSPH~1\COMMER~1\lib\ibmjceprovider.jar;C:\PROGRA~1\WEBSPH
~
1\COMMER~1\lib\local_policy.jar;C:\PROGRA~1\WEBSPH~1\COMMER~1\lib\US_export_pol
i
cy.jar;;;C:\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\WCSCom
m
on-ejb.jar;C:\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\WCSCF
u

```

```

lfillment-ejb.jar;;C:\WebSphere\AppServer\classes\ivjfix.jar;;C:\WebSphere\AppS
e
rver\installedApps\WC_Enterprise_App_demo.ear\lib\wscmcruntime.jar;;C:\WebSpher
e
\AppServer\installedApps\WC_Enterprise_App_demo.ear\lib\wcsdatabasean.jar;;C:\PRO
G
RA~1\WEBSPH~1\COMMER~1\lib\wcsruntime.jar;;C:\WebSphere\AppServer\installedApps
/
WC_Enterprise_App_demo.ear\lib\wscatalog.jar;;C:\WebSphere\AppServer\installed
A
pps\WC_Enterprise_App_demo.ear\WCSServer-ejb.jar;;C:\WebSphere\AppServer\instal
l
edApps\WC_Enterprise_App_demo.ear\WCSMCPProductAdvisor-ejb.jar;;C:\WebSphere\App
S
erver\installedApps\WC_Enterprise_App_demo.ear\WCSCatalog-ejb.jar;;C:\WebSphere
\
AppServer\installedApps\WC_Enterprise_App_demo.ear\lib\wcslogging.jar;;C:\WebSp
h
e\AppServer\lib\j2ee.jar;;C:\WebSphere\AppServer\lib\ns.jar;;C:\WebSphere\App
S
erver\installedApps\WC_Enterprise_App_demo.ear\lib\wcssfc.jar;;C:\WebSphere\App
S
erver\installedApps\WC_Enterprise_App_demo.ear\properties;C:\PROGRAM~1\WEBSPH~1\
C
OMMER~1\lib\sslite.zip;C:\Program Files\SQLLIB\java\db2java.zip;;C:\Program
File
s\SQLLIB\java\db2java.zip;C:\Program Files\SQLLIB\java\runtime.zip;C:\Program
Fi
les\SQLLIB\java\sqlj.zip;C:\Program Files\SQLLIB\bin;./"
-Dcom.ibm.CORBA.ConfigU
RL=file:/C:\WebSphere\AppServer\properties\sas.client.props
com.ibm.commerce.pa.
admin.PABatchXML -t C:\PROGRAM~1\WEBSPH~1\COMMER~1\xml\tools\pa\pabatch.dtd -x
ml
0101.xml -c demo.xml -d 1
<<<<---PABatchXML Started --->>>>

+++++
Product Advisor Batch Utility
Licensed Materials - Property of IBM
5724-A18
(c) Copyright IBM Corp. 1998, 2001.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
+++++
instance: demo
java.naming.provider.url: iiop://localhost:900
BuildMetaphors is initialized
buildDOMTree <<<<

```

```

1.1.16
buildDOMTree >>>>
processStoreBegin <<<
processStoreBegin: SID=10101
-----success in new store-----SID=10101
processStoreBegin >>>
processCategoryBegin <<<
processCategoryBegin: ID=10101
verifyCategoryID <<<
verifyCategoryID >>>
verifyCategoryID <<<
verifyCategoryID >>>
-----success in set catID in new merchant-----ID=10101
processCategoryBegin >>>
processPCBegin <<<
/webapp/wcs/stores/servlet/pc51.jsp
Size
1
1
1
-----
PARTNUMBER
8
2
1
-----
THUMPNAIL
9
1
1
-----
SHORTDESCRIPTION
10
1
1
-----
buildProductComparer <<<
LogConfiguration.loadConfiguration: no logging configuration found (Null XML
Nod
e). Standard output is used for logging error messages.
Metaphor add ok!
buildProductComparer >>>
processPCBegin >>>
processCategoryEnd <<<
processCategoryEnd 10101
processCategoryEnd >>>
processStoreEnd <<<
processStoreEnd -1
processStoreEnd >>>

```

```
BuildMetaphors is completed
<<<<---PABatchXML Finished!--->>>>
```

For more information about creating Product Advisor input files refer to the WebSphere Commerce Version 5.4 Online Help.

13.1.4 Testing product comparison

A product comparison sample JSP file is located in the directory
<WCS_DIR>\CommerceServer\samples\web\pa\pc51.jsp

To test the product comparison data we used this sample JSP file. The steps are:

1. Copy the sample JSP to the directory where the sample store is published.
For example:

```
<drive>\WebSphere\AppServer\installedApps\WC_Enterprise_App_instance
_name.ear\wcstores.war\WEB-INF\classes\storedir
```

2. Open the pc51.jsp and search the following code:

```
<jsp:setProperty property="orientation" name="pcTable"
value="VERTICAL"/>
```

3. Change the orientation property value to get a list in the horizontal position:

```
<jsp:setProperty property="orientation" name="pcTable"
value="HORIZONTAL"/>
```

4. Save the file and close it.
5. Launch to the following URL:

```
https://local_name/webapp/wcs/stores/servlet/<store>7pc51.jsp?storeId=store_
id&categoryId=category_id&langId=-1
```

For our example we accessed the sample JSP with the following URL:

```
https://local_name/webapp/wcs/stores/servlet/WebFashion/pc51.jsp?storeId=101
01&categoryId=10101&langId=-1
```

You will get a list of all products and items with the same attributes as shown in Figure 13-1.

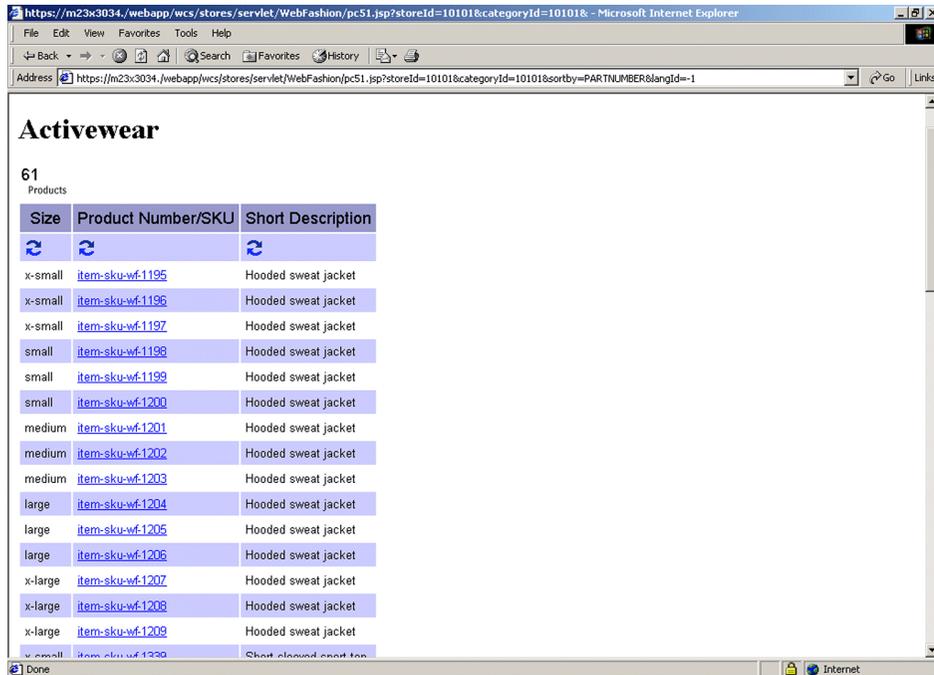


Figure 13-1 Product comparison sample JSP

You can sort the list by Size, Product Number and Short Description.

For more information about product comparison and the other Product Advisor metaphors refer to the WebSphere Commerce Version 5.4 Online Help, the *IBM WebSphere Commerce Fundamental Version 5.4* or to the *WebSphere Commerce Accelerator Customization Guide Version 5.4*.

13.2 Products and bundles

In this section we discuss product creation and customization. We also describe how to implement and display product bundles.

13.2.1 Definition of products

Several types of catalog entries classify merchandise in the WebSphere Commerce database. One of these types is the product.

A product is a group of items that exhibit the same attributes. For example a t-shirt. The attributes of this product are color and size, but a small t-shirt with colors in orange and black is an item. However, items do not need to be related to any product, and can exist independently in the catalog.

The relationship between products and items is in the CATENTREL table.

The product management tools in the WebSphere Commerce Accelerator allows you to manage the products in your store's master catalog using various wizards and notebooks.

In the WebSphere Commerce Accelerator using the product manager you can create or set characteristics:

- ▶ The product code, which uniquely identifies the product.
- ▶ The product name and description.
- ▶ The merchandising options, such as indicating that a product displays to customers, or that it is part of a special promotion.
- ▶ Thumbnail and full size images of the product.
- ▶ Tax specifications and shipping specifications.
- ▶ Discounts assigned to the product.

13.2.2 Creating new product with product manager

InFashion is a business-to-consumer online fashion store provided with WebSphere Commerce. The sample store is packaged with WebSphere Commerce as a store archive, and as a result, no further installation is necessary. All that is required to view the sample store is to create a new store archive based on InFashion using the Store Services tools, then publish it to the WebSphere Commerce Server.

In this section we create a new product for an existing category in the InFashion store. If you want to create a new new product in other sample stores refer to the WebSphere Commerce Version 5.4 Online Help.

For more information of creating stores from and existing store archive, see “Creating a store archive using Store Services” in the WebSphere Commerce Version 5.4 Online Help.

Create a new product

In the first step we create a new product with the WebSphere Commerce Accelerator:

1. Choose **Merchandise -> Products**. A list of products for the store displays.
2. Click **New**. The general page is displayed.

3. Add the following details to the product. See Figure 13-2
 - ▶ Product Code : 1234
 - ▶ Name : Special t-shirt
 - ▶ Display to customers: enabled
4. Select the **For purchase** check box to specify that customers can include the product in their shopping carts and order the product

The screenshot shows a web browser window titled "WebSphere Commerce Accelerator - Microsoft Internet Explorer". The address bar shows "InFashionFFM - Mystore - United States English" and the breadcrumb "Logout > Home > Products > New Product". The main content area is titled "General" and contains a list of categories on the left: General, Description, Category, Images, Manufacturer, Discounts, Sales Tax, Shipping Tax, Shipping, Categories, Units of Measure, Fulfillment, and Advanced. The "General" category is selected. The form fields are as follows:

Product Code (required)	<input type="text" value="1234"/>
Name (required)	<input type="text" value="Special t-shirt"/>
Display to customers	<input checked="" type="checkbox"/>
For purchase	<input checked="" type="checkbox"/>

At the bottom right of the form, there are two buttons: "Next" and "Cancel".

Figure 13-2 Create a new product

5. Click **Next**. Enter the following information in the description page:
 - ▶ Short description: Special t-shirt
 - ▶ Long description: Skater style, lightweight t-shirt, Crew neck, 80 % cotton, 20 % polyester, Machine-washable.
6. Click **Next**. In the category page select the appropriate parent category for your product as shown in Figure 13-3. We placed our product in the shirt category

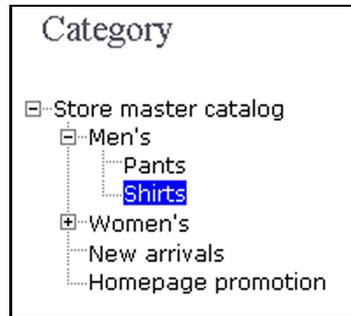


Figure 13-3 Select parent category for the product

7. Click **Next**.

In the images page in the full size image file and location field, type the full or relative path to the image, including the image name. We used images/mens_activewear_ultimate.gif

In the thumbnail image file and location field, type the full or relative path to the image, including the image name. We used images/mens_activewear_ultimate_sm.gif

This directory is where the store images are located on the WebSphere Commerce machine. For our example we used images that are included in the InFashion sample store.

8. Click **Next**. The manufacturer information page is displayed. Enter the following information:

- ▶ Manufacturer Part Number: 1234
- ▶ Manufacturer Name: Infashion

9. Click **Next**. The discounts page displays. You can apply discounts to the product.

10. Click **Next**. The sales tax page displays. Select a tax name from the available taxes list, and click **Add**. If there is no tax available in the list a message appears.

11. Click **Next**. The shipping tax page displays. Select the tax name from the available taxes list, and click **Add**. If there is no shipping tax available in the list a message appears.

12. Click **Next**. The shipping Categories page displays. Select a category name from the available categories list, and click **Add**. A shipping category is defined by a range in weight, size or amount. If no shipping category is available, a message appears.

13. Click **Next**. The units of measure for shipping page displays. Supply values for the following :

- ▶ Sold in Multiples of This Amount field: 1.
- ▶ Number of Items per package field: 1
- ▶ Unit of Measure drop-down list: one

14. Click **Next**. The product fulfillment page displays. Accept the default values:

- ▶ Track inventory: enable checkbox
 - ▶ Allow backup: enable checkbox
 - ▶ Force backorder: disable checkbox
 - ▶ Release separately: disable checkbox
 - ▶ Returnable: enable checkbox
1. Creditable: enable checkbox
 1. Smallest Amount that can be measured field: 1.
2. Click **Next**. The advanced page displays. We do not need to enter any information for our example.
 3. Click **Finish**. The changes will be saved.

Adding a price to a new product

The price of a product can be defined for one or more currencies. In our example the currencies used are USD and EUR. You have to define prices for all currencies. Use the WebSphere Commerce Accelerator:

1. Choose **Merchandise -> Products**. A list of products for the store is displayed.
2. Select the check box to the left of the product 1234 and click **Prices**
3. Enter 29.00 for the USD price and 29.00 for EUR price. See Figure 13-4
4. To save the price click **OK**.

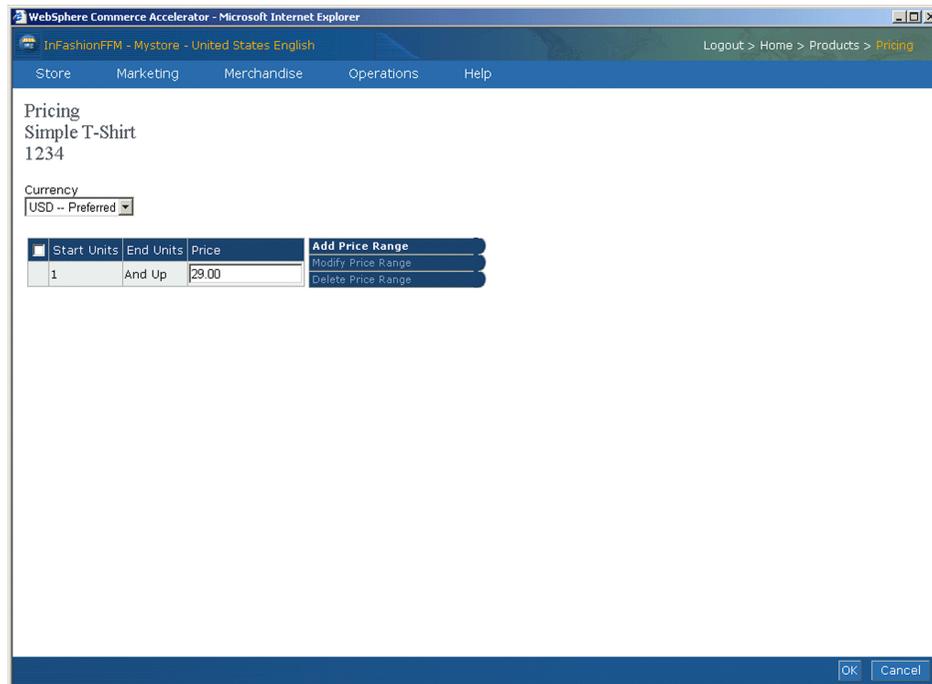


Figure 13-4 Adding a new price

Adding new attributes and values

You have to add attributes such as color or size to the new product. These are typical attributes for a shirt. Each combination of attributes and attribute values equals a new SKU.

1. Choose **Merchandise -> Products**.
2. Select the check box to the left of the product 1234 and click **Attributes**.
3. Click **New**. The new attribute page displays.
4. Select the default language from the drop-down list: United States English
5. Enter the following information in the new attribute page:
 - ▶ Insure that the attribute type Text is selected:
 - ▶ Name : Size
 - ▶ Description : Size
 - ▶ In the value field type x-small, click **Add**.
 - ▶ In the value field type small, click **Add**.
 - ▶ In the value field type medium, click **Add**.
 - ▶ In the value field type large, click **Add**.
 - ▶ In the value field type x-large, click **Add**.

See Figure 13-5 for an example.

6. Click **OK**. You will get a message after creating the attribute

WebSphere Commerce Accelerator - Microsoft Internet Explorer

InFashionFFM - Mystore - United States English Logout > Home > Products > Simple T-Shirt - Attributes > Change Attribute

Store Marketing Merchandise Operations Help

Change Attribute

Language
United States English

Name (Required)
Size

Description
Size

Type
 Text Whole Number Decimal Number

<input checked="" type="checkbox"/>	Reference Value	Value	Image (pathname)	Sequence	Add Value
<input type="checkbox"/>	x-small	x-small		1.0	Change Value
<input type="checkbox"/>	small	small		2.0	Move Up
<input type="checkbox"/>	medium	medium		3.0	Move Down
<input type="checkbox"/>	large	large		4.0	Delete Value
<input type="checkbox"/>	x-large	x-large		5.0	

OK Cancel

Figure 13-5 Adding attribute size to a new product

7. Create the second attribute . Click **New**. Enter the following information in the new attribute page:

- ▶ Name : Color
- ▶ Description : Color
- ▶ In the value field type black/white, click **Add**.
- ▶ In the value field type green/navy, click **Add**.
- ▶ In the value field type orange/black, click **Add**.

See Figure 13-6 for an example.

8. Click **OK**. You will get a message after creating the attribute.

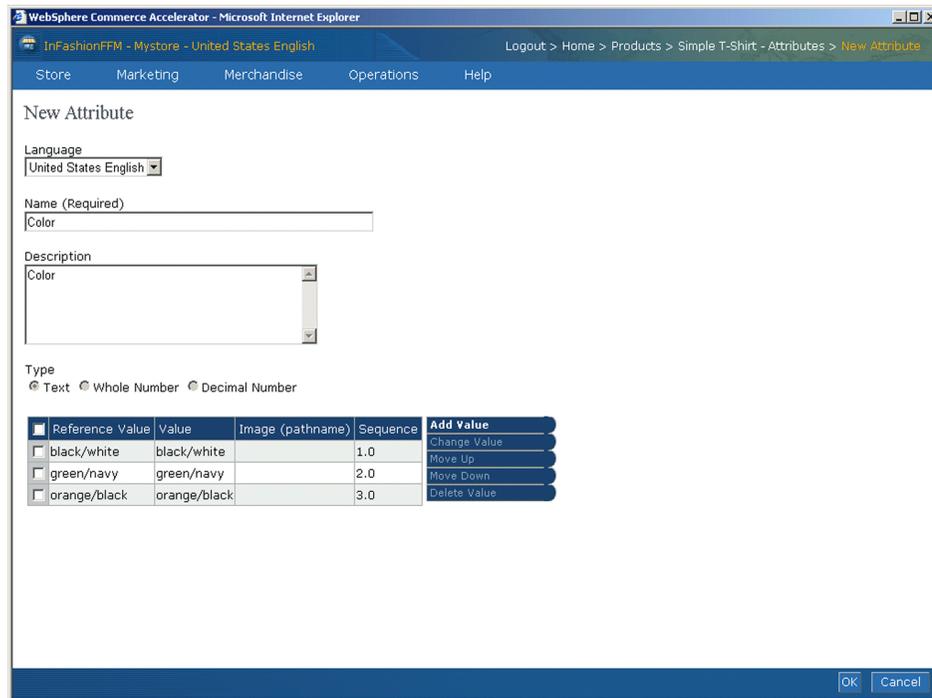


Figure 13-6 Adding attribute color

Generate SKUs for the new product

After creating the attributes and the attribute values for the product you are allowed to generate SKUs. To generate SKUs for a product, do the following:

1. Choose **Merchandise -> Products**. A list of products for the store displays.
2. Select the check box to the left of the product 1234 and click **Generate SKUs**
It takes a few minutes to create the SKUs. For the t-shirt, fifteen different SKUs will be created .
3. After the SKUs have generated successfully, click **SKUs** to list the newly created SKUs for your product. See Figure 13-7 for an example.

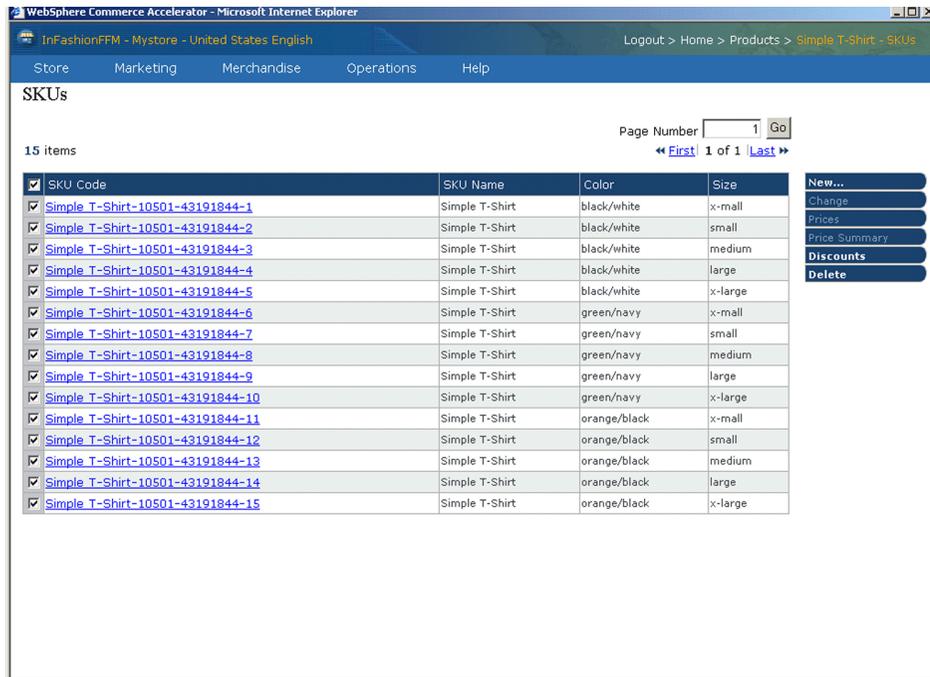


Figure 13-7 Created SKUs

13.2.3 Testing the new product

To show the new product in the sample store:

1. You can launch your store by entering the following URL:

`https://<host_name>/webapp/wcs/stores/store_directory/index.jsp`

2. Click the **Men's** link on the top navigation bar and select the **Shirts** link. The new product Special t-shirt appears on the categorie page with the new price \$ 29, as shown in Figure 13-8

As an alternative to navigating through the store you can go directly to catalog display by entering a URL for the CatalogDisplay.jsp. The URL is:

`https://<hostname>/webapp/wcs/stores/servlet/CatalogDisplay?catalogId=catalog_id&storeId=store_id&category_Id=category_id&langId=-1`

For our example the URL used was:

`https://<hostname>/webapp/wcs/stores/servlet/CatalogDisplay?catalogId=10001&storeId=10001&category_Id=10005&langId=-1`

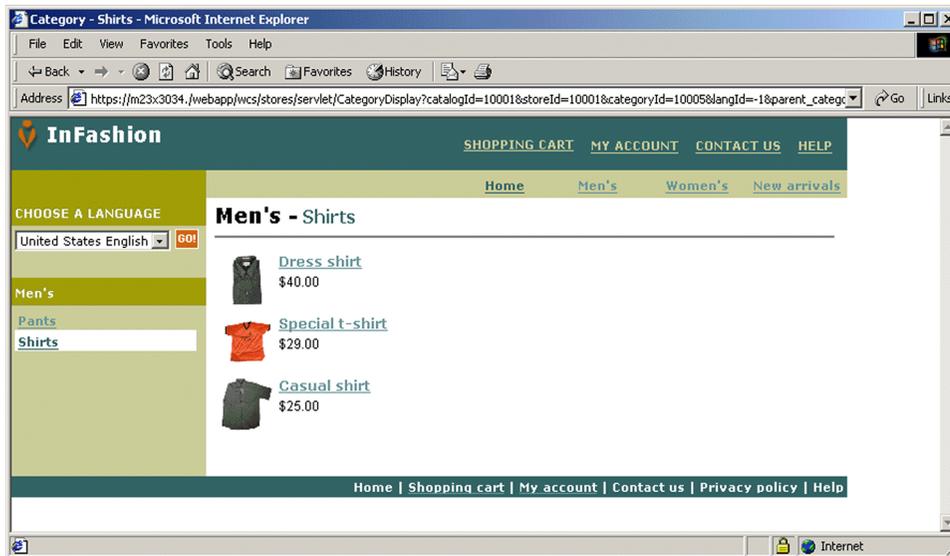


Figure 13-8 Catalog page with new product

3. Click on the **Special t-shirt** link and the product display page appears as shown in Figure 13-9. You will see the new product with all the attribute values for color and size.

As an alternative to navigating through the store you can go directly to product display by entering a URL for the ProductDisplay.jsp. The URL is:

```
https://<hostname>/webapp/wcs/stores/servlet/ProductDisplay?catalogId=catalog_id&storeId=store_id&product_Id=product_id&langId=-1
```

For our example the URL used was:

```
https://<hostname>/webapp/wcs/stores/servlet/ProductDisplay?catalogId=10001&storeId=10001&product_Id=10501&langId=-1
```

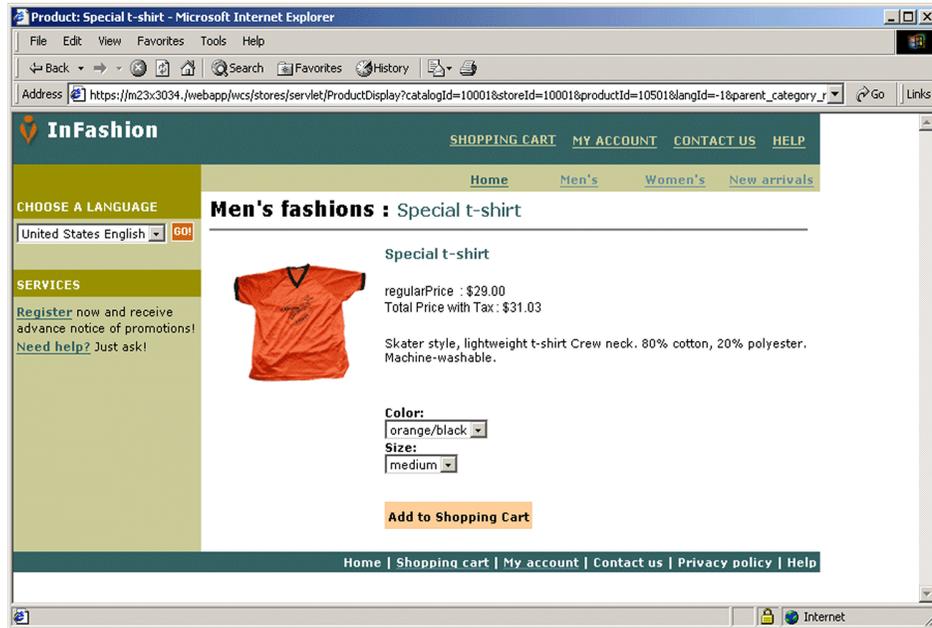


Figure 13-9 Product display page for the new product

13.2.4 Definition of bundles

A bundle is a collection of catalog entries that allows customers to buy multiple items with a single click. For example, a bundle for a casual ensemble might be composed of a shirt, pants and a belt.

A bundle is a grouping of items, or a combination of products, items, and fully resolved packages. If you select a bundle which only contains items, the bundle is decomposed into separate SKUs that are added individually to the shopping cart. However, if you select a bundle which contains products, these products need to be resolved into items through SKU resolution before they can be added to a shopping cart. In either case, once a bundle is decomposed and its component items are added to a shopping cart, you can modify or remove each item. The total price of the bundle is the total price of all the bundle components.

13.2.5 Bundles in WebFashion

WebFashion is a business-to-consumer online fashion stores provided with WebSphere Commerce. One of the features included in the WebFashion store is bundles. A bundle typically includes:

- ▶ Description

- ▶ List of components
- ▶ Price for each component
- ▶ Image
- ▶ List of attributes (for example color or size)
- ▶ Values for the attributes (for example, blue or small, large)

To create new bundles, you have to added new information to the catalog.xml file. The catalog.xml file stores the catalog information for the sample stores in WebSphere Commerce.

To use the WebFashion sample store for bundle customizations, first create a new archive based on WebFashion using the Store Services tools. You can add the bundle changes in the catalog.xml before publishing the store . If you have already published the store you can use the loader package or the Store Services to publish store database assets.

For more information how to create a store archive and publish the store database assets refer to the WebSphere Commerce Version 5.4 Online Help.

13.2.6 Creating bundles

After you have created products and items for your catalog, you must create the bundles. Begin creating bundles by adding information to the CATENTRY and CATENTDESC tables. Then create the relationships between bundles and their components to the CATENTREL table. Figure 13-10 shows the data model for the WebSphere Commerce catalog

For example, we create a bundle which is composed of plain front cotton pants and a classic belt. The name of the bundle is modern combination.

catalog.xml	08.05.2002 1...	1.319.586	96%	54.588	data\
catalog.xml	08.05.2002 1...	1.041.554	95%	53.065	data\es_US\
catalog.xml	08.05.2002 1...	1.019.129	95%	52.846	data\en_US\

Figure 13-11 catalog.xml in store archive

4. Add the information shown in Example 13-7 into the catalog.xml to create a new base item. Base items represent a general family of products with a common name and description.

Example 13-7 New base item

```

<baseitem
baseitem_id="@baseitem_id_5001"
member_id("&MEMBER_ID;")
markfordelete="0"
partnumber="webfashion_5001"
itemtype_id="ITEM"
quantitymeasure="C62"
quantitymultiple="1.0"
/>

```

The baseitem_id is the generated unique key.

The partnumber uniquely identifies the base item for the owner.

5. Add the information shown in Example 13-8 to create the relationship between the item version and the base item in the WebSphere Commerce database:

Example 13-8 Relationship of item version and base item

```

<itemversn
itemversn_id="@itemversn_id_5001"
baseitem_id="@baseitem_id_5001"
expirationdate="2010-01-01 00:00:00.000000"
versionname="version"
/>

```

The itemversn_id is a generated reference number which identifies the item version.

The baseitem_id is the base item.

6. Add the distribution arrangements into the catalog.xml. A distribution arrangement enables a store to sell its own inventory. See Example 13-9

Example 13-9 Distribution arrangement

```

<distarrang

```

```
distarrang_id="@distarrang_id_5001"
wholesalestore_id="@storeent_id_1"
merchantstore_id="@storeent_id_1"
baseitem_id="@baseitem_id_5001"
pickingmethod="F"
startdate="2000-12-25 00:00:00.000000"
enddate="2010-01-01 00:00:00.000000"
/>
```

The `distarrang_id` is the reference number of the distribution arrangement.

The `baseitem_id` is the product covered by the distribution arrangement.

7. Add the information shown in Example 13-10 to the `catalog.xml` to get information for the store items.

Example 13-10 New store item

```
<storeitem
baseitem_id="@baseitem_id_5001"
storeent_id="@storeent_id_1"
trackinventory="Y"
forcebackorder="N"
releaseseparately="N"
returnnotdesired="N"
backorderable="Y"
creditable="Y"
minqtyforsplit="0"
/>
```

The `baseitem_id` is the base item.

The `storeent_id` is the store or the store group.

8. Search for Bundle. The `catalog.xml` contains an example of a bundle. Add the code shown in Example 13-11 to create a new catentry for the new bundle.

Example 13-11 New catentry

```
<!--Bundle2-->
<catentry
catentry_id="@product_id_5001"
baseitem_id="@baseitem_id_5001"
member_id="&MEMBER_ID;"
catenttype_id="BundleBean"
partnumber="item-sku-wf-5001"
mfpartnumber="item-sku-wf-5001"
mfname="webfashion"
markfordelete="0"
buyable="1"
```

```
/>
```

Each time you create a bundle , the `catentry_id`, `partnumber`, and `mfpartmentnumber` change to create different bundles. In our example for the new bundle the values were:

```
catentry_id="@product_id_5002",
partnumber="item-sku-wf-5002",
mfpartmentnumber="item-sku-wf-5002"
catentrytype_id="BundleBean" to identify the entry as a bundle
```

9. After creating a new `catentry`, define the relationship between a catalog group and catalog entry by adding information to the `CATGRPPENREL` table as shown in Example 13-12

Example 13-12 Relationship between catalog group and catentry

```
<catgpenrel
catgroup_id="@catgroup_id_11"
catalog_id="@catalog_id_1"
catentry_id="@product_id_5001"
sequence="0"
/>
```

The `catgroup_id_11` is the men's fashions category

10. After creating the relationship between the catalog group and `catentry`, you have to define the relationships of the bundle and its components, by adding information to the `CATENTREL` table:

The `product_id_1660` is for plain front cotton pants.

The `product_id_102` is for the classic belt.

These are the components of the new bundle. Example 13-13 shows the XML definition we used.

Example 13-13 relationship between catalog group and catentry

```
<catentrel
catentry_id_parent="@product_id_5001"
catreltype_id="BUNDLE_COMPONENT"
catentry_id_child="@product_id_1660"
sequence="1"
quantity="1"
/>
```

```
<catentrel
catentry_id_parent="@product_id_5001"
catreltype_id="BUNDLE_COMPONENT"
catentry_id_child="@product_id_102"
sequence="2"
```

```
quantity="1"  
</>
```

11. Add the codes shown in Example 13-14 to the catalog.xml to get information about the base item for the STORITMFFC table.

Example 13-14 Storitmffc definition

```
<storitmffc  
baseitem_id="@baseitem_id_5001"  
storeent_id="@storeent_id_1"  
ffmcenter_id="@ffmcenter_id_1"  
shippingoffset="86400"  
</>
```

12. Save the modified catalog.xml file.

13. Next add the bundle description to the CATENDESC table in the locale-specific catalog.xml file for example for the English language. Open in the archive the catalog.xml under the folder data\en_US.

Example 13-15 shows the code to add the description of the bundle

Example 13-15 Adding description of the bundle into the catalog

```
<!--Bundle-->  
<catentdesc  
catentry_id="@product_id_5001"  
language_id="&en_US;"  
name="Special Bundle"  
shortdescription="Modern combination"  
longdescription="This pant and belt combination is perfect for those casual  
days at the office."  
fullimage="images/mens_pants_grey.gif"  
available="1"  
published="1"  
</>
```

14. Add the base item description to the same XML file as shown in Example 13-16

Example 13-16 Adding base item description

```
<baseitmdsc  
baseitem_id="@baseitem_id_5001"  
language_id="&en_US;"  
shortdescription="Special Bundle"  
longdescription="This pant and belt combination is perfect for those casual  
days at the office."  
</>
```

15. Save the file and close it. Repeat steps 13 and 14 for any second language supported by your store. For example WebFashion supports Spanish.
16. Add catalog entries to the store-catalog relationship in the store-catalog.xml as shown in Example 13-17

Example 13-17 Add catalog entries to the store-catalog relationship

```
<!--bundle-->
<storecent
storeent_id="@storeent_id_1"
catentry_id="@product_id_5000"
/>
```

The storeent_id is the reference number of the store entity in the database.

The catentry_id is the reference number of the catalog entry.

17. Save and close the file.
18. From your working directory packages the following files to build a new store archive with a ZIP program. The name of the new store archive must be the same as your original store archive, for example WebFashion.sar.
 - data
 - SAR-INF
 - proorties.zip
 - webapp.zip
19. Copy the store archive WebFashion.sar into the SAR directory replacing the original SAR. The director used by SAR files is
<WCS_DIR>\CommerceServer\instances\instancename\sar
20. Publish only the store database assets with Store Services or use the Loader package to import the XML files. For more informations how to load data into the database refer to Chapters 27 and 28 in the *IBM WebSphere Commerce Store Developer's Guide Version 5.4*, or to the WebSphere Commerce Version 5.4 Online Help.

Testing the new bundle

To show the bundle product in the sample store :

1. You can launch your store by entering the following URL:
`https://<host_name>/webapp/wcs/stores/store_directory/index.jsp`
2. Click the **Men's** link on the top navigation bar. A new bundle is displayed as shown in Figure 13-12.

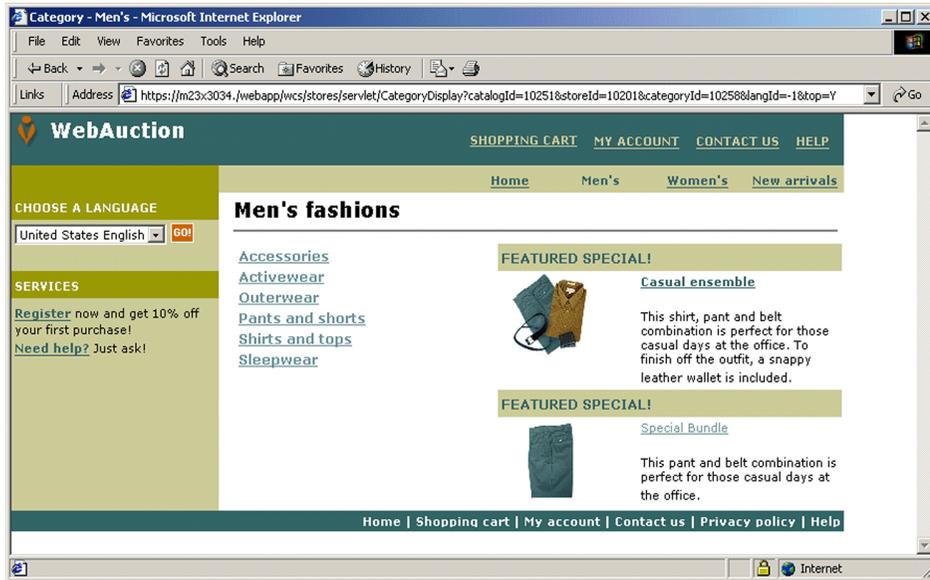


Figure 13-12 New bundle

3. Click **Special Bundle** and you will get the details of the new bundle as shown in Figure 13-13.

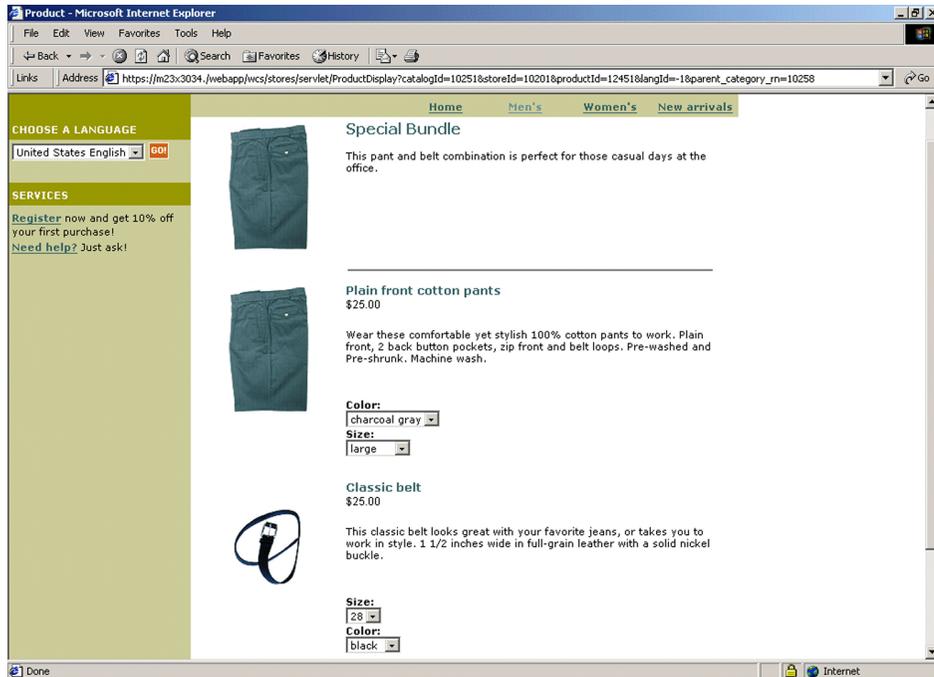


Figure 13-13 Special bundle

13.3 Display of multiply currencies

For a site with multiple stores, you can use different currencies for each store, or you can assign currencies to the store group. For each store you can define how the currency is displayed.

Customers are allowed to select a shopping currency. All amounts on the store pages are then displayed in the selected currency.

The stores in the WebSphere Commerce samples display prices in one currency, but it is also possible to display the prices in two different currencies. The prices for the items that they have added to the shopping cart and the total order price are automatically converted, recalculated and displayed in both currencies as shown in Figure 13-14.

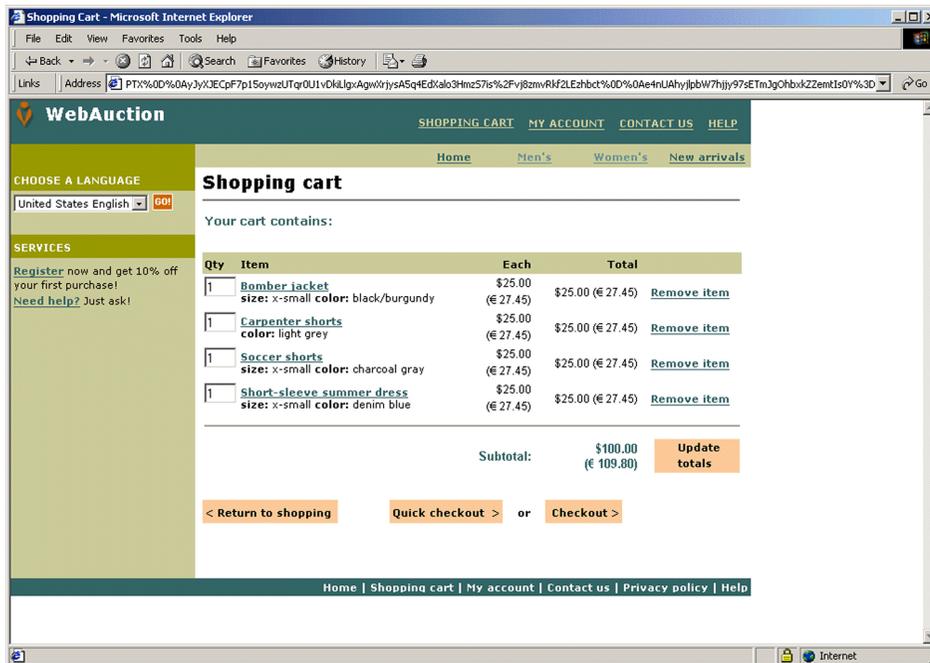


Figure 13-14 Shopping cart with dual display of currencies (USD and EUR)

13.3.1 Currencies types

WebSphere Commerce stores support several different currency types. These types include:

1. Default currency

Each store, or group of stores, has a default currency. The Store Services tools in WebSphere Commerce allow you select a default currency and to add supported currencies to the store. The default currency for a store is specified in the STOREENT table. The SETCURR column contains the default currency for a store which will be used by a customer that has not specified a preferred currency. If it is NULL for a store, the default currency is obtained from the store group. See also the STORELANG table.

2. Supported currencies

A store can have many supported currencies. A supported currency is one in which payment is accepted.

Table 13-2 shows the ISO 4217 codes for each international currency currently supported by WebSphere Commerce. The codes appear in the SETCCURR column of the SETCURR database table.

Table 13-2 Supported currencies in WebSphere Commerce

Country/Region	Currency name	ISO 4217 Code
Argentina	Argentine peso	ARP
Austria	Austrian schilling	ATS
Australia	Australian dollar	AUD
Belgium	Belgian franc	BEF
Brazil	Brazilian real	BRL
Canada	Canadian dollar	CAD
China	Chinese yuan renminbi	CNY
Euro-supported countries: Austria Belgium Finland France Germany Ireland Italy Luxembourg Netherlands Portugal Spain	European euro currency	EUR
European currency unit-supported countries	European currency unit. This unit is not the same as the euro currency.	XEU
Finland	Finnish marka	FIM
France	franc	FRF
Germany	Deutsche mark	DEM
Hong Kong S.A.R. of China	Hong Kong dollar	HKD
Ireland	Irish punt	IEP
Italy	Italian lira	ITL
Japan	Japanese yen	JPY
Korea, Republic of (South Korea)	South Korean won	KRW

Country/Region	Currency name	ISO 4217 Code
Luxembourg	Luxembourg franc	LUF
Netherlands	Dutch guilder	NLG
Portugal	Portuguese escudo	PTE
Singapore	Singapore dollar	SGD
South Africa	South African rand	ZAR
Spain	Spanish peseta	ESP
Switzerland	Swiss franc	CHF
Taiwan	Taiwanese dollar	TWD
United Kingdom	British pound	GBP
United States	United States dollar	USD

3. Shopping currency

The shopping currency is the currency in which customers pay for products in the store. All monetary amounts on the store pages are displayed in this currency.

If the customer's preferred currency is supported by the store, it then becomes the shopping currency

If the preferred currency is not supported at all, the default currency for the store or store group specified in the STOREENT table is used as the shopping currency.

If the store does not have a default currency specified in the STOREENT table for its store or store group, then the store's default currency for the customer's language ID is used. The STORELANG table determines that setting.

13.3.2 Dual display and counter values

The dual display of a prices allows simultaneous display of an amount in the shopping currency unit and in the another currency. You can enable the store pages to display equivalent prices in currencies other than the shopping currency. In our example we had USD as shopping currency and also displayed prices in EUR. The display of an equivalent price is called the counter value.

In our example we set the counter value currency to the EUR. The counter values are displayed in the store pages. They are specified in the CURCVLIST table. The counter value currency is only valid, if there is a conversion in the CURCONVERT table from the shopping currency to the counter value as shown in Figure 13-15.

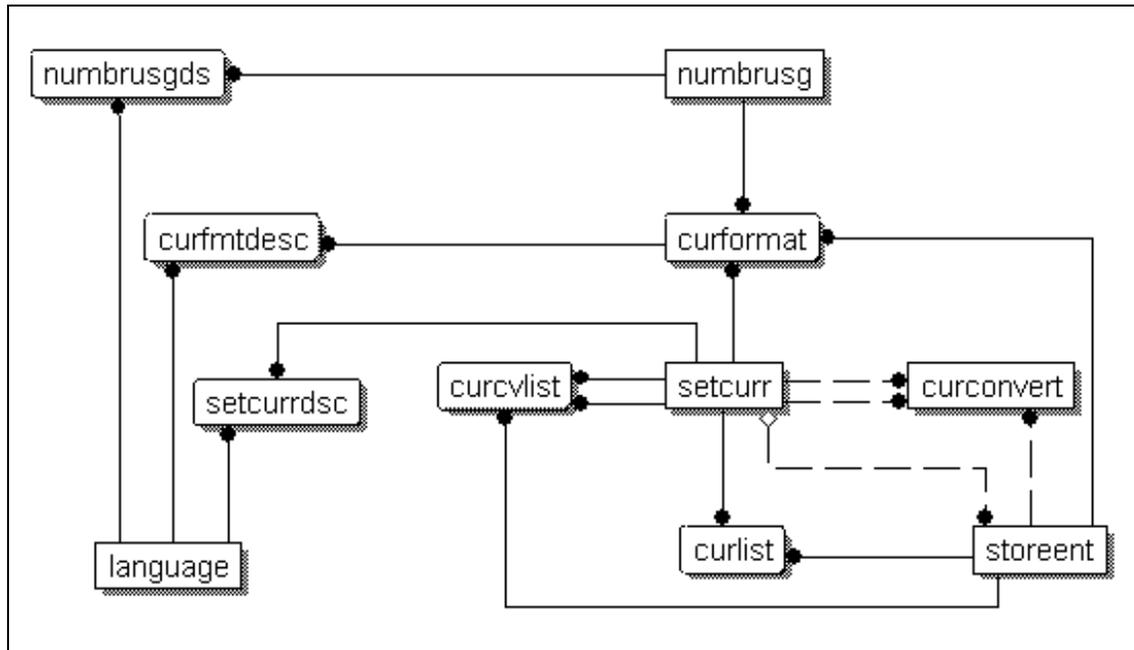


Figure 13-15 Currency data model

13.3.3 Implementation of dual display of currencies

WebFashion is a business-to-consumer online fashion store provided with WebSphere Commerce. The sample store is packaged with WebSphere Commerce as a store archive, and as a result, no further installation is necessary. All that is required to view the sample store is to create a new store archive based on WebFashion using the Store Services tools, then publish it to the WebSphere Commerce Server.

We changed the WebFashion sample store to display two different currencies.

For more information, see “Creating a store archive using Store Services” in the WebSphere Commerce Version 5.4 Online Help.

Enabling euro currency support

To enable the euro currency support in the sample store:

1. Ensure that the IBM WC Configuration Manager Server service has been started. This service is listed in the Windows services panel.
2. Choose **Start -> Programs -> IBM WebSphere Commerce -> Configuration**.
3. The configuration authentication window opens.
4. Enter a configuration manager user ID and password and click **OK**. The default user ID is webadmin and the default password is webibm.
5. Expand the instance and click on **Caching Subsystem**. Go to the Advanced tab and select the checkboxes for **Session Dependent Caching** as shown in Figure 13-16. Click **Apply**
6. Close the Configuration Manager.

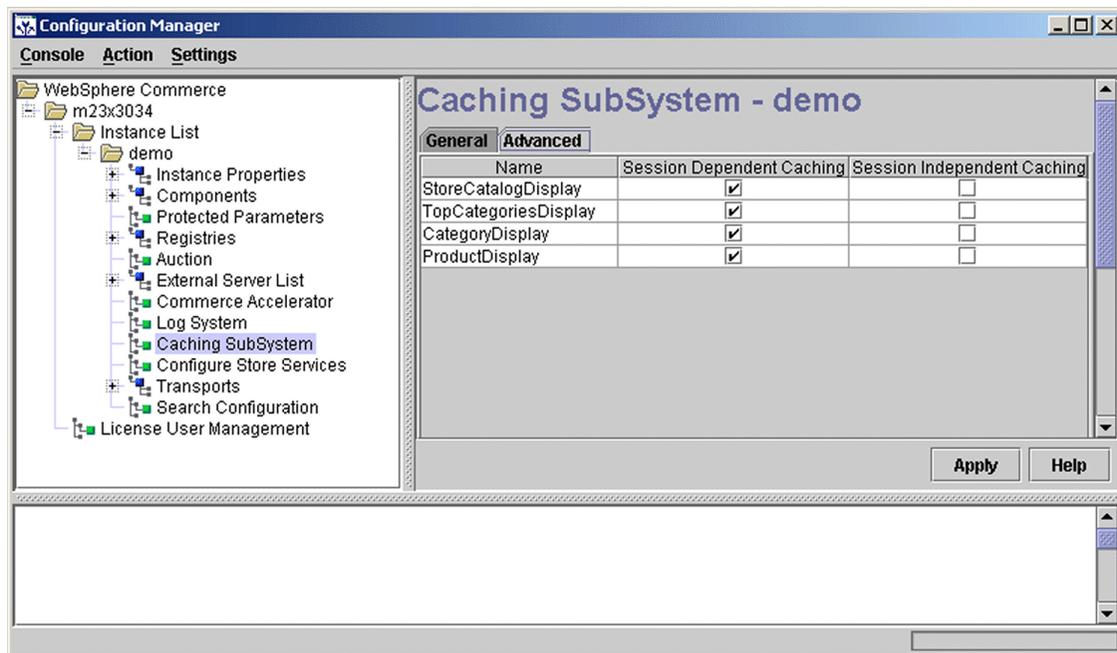


Figure 13-16 Configuration manager

Add supported currencies

After enabling the euro support you have to add supported currencies to the database into the table CURLIST. Follow the next steps:

1. Run the SQL command in a db2 command window as shown in the Example 13-19. In the SQL command the values are:
 - ▶ store_entity_id is the store or store group ID.

- ▶ currency is the 3 character ISO 4217 currency code representing the supported currency. This code must appear in the SETCCURR column of the SETCURR table.

Example 13-18 example for the storeid 10201 and currency EUR

```
insert into curlist (storeent_id,currstr)
values(10201,'EUR')
```

The store_entity_id for our example is 10201 and the currency is EUR.

2. Refresh the Currency Manager Registry. Open the Administration Console and log on as a site administrator.
3. Choose **Configuration -> Registry**
4. Select **Currency Manager** and click **Refresh** to reload the Registry window and check on the status of components you are updating. When updating is complete, the status column reads "Updated."

Enabling dual display of supported currencies

After adding supported currencies to the database you have to enabling the dual display of the supported currencies by adding informations into the CURCVLIST table. Each row of this table represents a CounterCurrencyPair. The primary use of this information is for "dual display". Follow the next steps:

1. Run the SQL Command in the db2 command window as shown in the example 13-20:

store_entity_id is the store or store group ID.

currency is the three character ISO 4217 currency code representing the currency. This code must appear in the SETCCURR column of the SETCURR table.

counter_value_currency

is the three character ISO 4217 currency code representing the counter value currency. This code must appear in the SETCCURR column of the SETCURR table.

display_sequence

is the number which indicates the presentation order of the counter value currency. Counter value currencies are displayed in ascending order based on the counter value display sequence specified in the DISPLAYSEQ column in the CURCVLIST table.

```
insert into curcvlist
(storeent_id,currstr,countervaluecurr,displayseq)
```

```
values(store_entity_id,'currency','counter_value_currency',displayseq
)
```

Example 13-19 example for enabling dual display of supported currencies

```
insert into curcvlist (storeent_id,currstr,countervaluecurr,displayseq)
values(10201,'USD','EUR',0.0)
```

The store_entity_id for this example is 10201, currency USD and countervalue_currency EUR.

2. Refresh the Currency Manger Registry. Open the Administration console and log on as a Site Administrator.
3. From the **Configuration** menu, click **Registry**
4. Select **Currency Manager** and click **Refresh** to reload the Registry window and check on the status of components you are updating. When updating is complete, the status column reads "Updated."
5. Close the Administration console
6. Remove temporary pages from the cache.

Verify the currency conversion

To display the prices in the supported currency there must be a conversion rate in the store. The conversion rate convert from the default currency (USD) into the supported currency (EUR):

The conversion informations are located in the CURCONVERT table . Review in the table if there exists a row for your store with a conversion rate. If not run the SQL Command in the db2 command window as shown in the example 13-21:

store_entity_id is the store or store group ID.

tocurr is the 3 character ISO 4217 currency code representing the converted currency. This code must appear in the SETCCURR column of the SETCURRE table.

conversion_factor is the conversion rate.

multiplyordivide whether to multiply ('M') or divide ('D').

bidirectional indicates whether the conversion is bi-directional ('Y') or unidirectional ('N').

updateable whether the conversion rate can be changed ('Y') or not ('N').

```

insert into curconvert
(storeent_id,fromcurr,tocurr,factor,multiplyordivide,bidirectional,u
pdatable,curconvert_id)
values(store_entity_id,'fromcurr','tocurr',conversion_factor,'M','Y
','Y',curconvert_id)

```

Example 13-20 example for the currency conversion

```

insert into curconvert
(storeent_id,fromcurr,tocurr,factor,multiplyordivide,bidirectional,updatabl
e,curconvert_id)
values(10201,'USD','EUR',1.0980400000000000000,'M','Y','Y',10201)

```

Verify shopping currency

The default currency for the store or store group specified in the STOREENT table is used as the shopping currency. Check if there exists in the column SETCC URR for your store a default currency. If there exists no default currency, make an update for your store in this table and set the value of SETCCURR to “USD”.

13.3.4 Testing the dual display of currencies

To show the dual display of the currencies in the sample store follow the next steps:

1. You can launch your store by entering the following:

```
https://host_name/webapp/wcs/stores/store_directory/index.jsp
```

2. Click the **Men's** link on the top navigation bar. Click on the categorie link **Pant and short**. The categorie page is displayed with the shopping currency USD and the counter currency EUR as shown in Figure 13- 17.

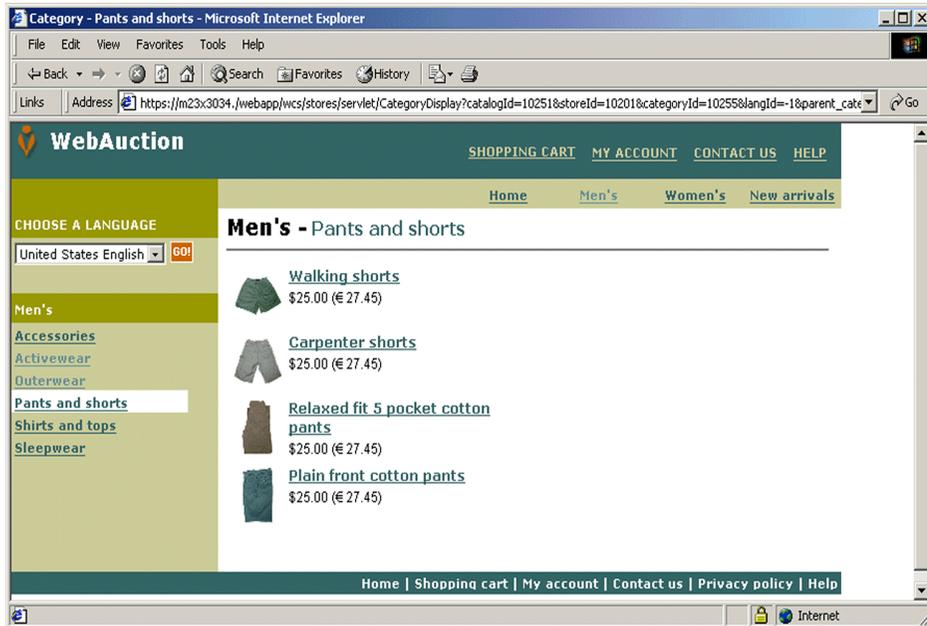


Figure 13-17 Category page with dual display

3. Click the **Carpenter Shorts** and you will get the details of the product with dual display as shown in Figure 13-18.



Figure 13-18 Product display page with dual display



Migration examples

WebSphere Commerce Business Edition Version 5.4 introduces some changes to the commerce environment from past releases. Store developers and administrators need to be aware of these changes in order to effectively create and deploy their stores. This chapter serves to outline some of the more important updates to the product and its underlying infrastructure and to provide an example of migration considerations.

14.1 Migration Overview

WebSphere Commerce Version 5.1 was a revolutionary change in IBM's commerce software. In a departure from past releases, WebSphere Commerce Version 5.1 was based entirely on a Java programming model. The move to Java brought with it all of the benefits of coding to an open industry standard.

While not nearly as drastic a change from the proceeding version as WebSphere Commerce Version 5.1, the release of WebSphere Commerce Business Edition Version 5.4 nonetheless introduces quite a few important updates. With an eye toward end of service dates and improvements in the WebSphere Commerce technology, this section highlights some of the more notable changes in the product.

This chapter is not intended to be an exhaustive review of product changes from WebSphere Commerce Version 5.1 to WebSphere Commerce Business Edition Version 5.4. Rather, we have elected to describe the updates which we expect to affect the largest number of implementations. For a complete review of product updates, please view the documentation available at the IBM WebSphere Commerce Business Edition Version 5.4 technical library at:

http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

Of particular interest are the following:

- ▶ *What's New in IBM WebSphere Commerce, 5.4 Version 5.4*
- ▶ *Migration Guide: IBM WebSphere Commerce*
- ▶ *IBM WebSphere Commerce Studio for Windows NT and Windows 2000 Migration Guide Version 5.4*
- ▶ *WebSphere Commerce Database Schema*
- ▶ WebSphere Commerce Version 5.4 Online Help

14.2 Migration considerations

In this section, we assemble and outline the WebSphere Commerce product changes that will affect the largest segment of the user population. The updates we review here fall into several high-level categories:

- ▶ Infrastructure changes
- ▶ Architecture changes
- ▶ Database changes

In addition, we examine some miscellaneous migration information that will assist developers and administrators with the product transition.

14.2.1 Infrastructure changes

There are changes to the underlying infrastructure environment which must be considered when migrating WebSphere Commerce. These include changes to WebSphere Application Server, the Web server running the environment and the database software.

WebSphere Application Server Version 4

The greatest platform difference for WebSphere Commerce Version 5.4 is the use of WebSphere Application Server Version 4 as the underlying application server engine. Earlier versions of WebSphere Commerce used WebSphere Application Server Version 3.5 and along with the new release comes compliance with the Sun's Java 2 Platform, Enterprise Edition (J2EE) specification. This causes several changes that must be accounted for in development and deployment of stores.

The use of a J2EE-compliant application server platform requires some changes in the way an application is packaged and deployed. Any code, commands and EJBs that have been customized in WebSphere Commerce Version 5.1 need to be redeployed to the level required for WebSphere Commerce Business Edition Version 5.4. Detailed information on this process can be found in *IBM WebSphere Commerce Studio for Windows NT and Windows 2000 Migration Guide Version 5.4*.

J2EE specifies a distinct separation between the development of an application and its administration. This leads to applications that are independent from the environment on which they are deployed which allows for simpler movement of code between environments as changes to the application code due to that movement are often not necessary. Rather, the deployment parameters associated with the applications change, a much easier undertaking.

An enterprise application in WebSphere Commerce Business Edition 5.4 can contain many Web modules and EJB modules. Each individual module can be installed on a separate application server or server group, including servers and server groups on multiple nodes. Conversely, a single application server or server group can have modules from many different applications on it. As you can no doubt imagine, there are many different ways to deploy a J2EE application.

Deployment of a new J2EE application is comprised of two steps: copying the appropriate files (Java classes, JSPs, static HTML and images files, etc) into the archive and creating the deployment descriptor files for the modules and the application. During the WebSphere Commerce migration, all migrated Web resources and enterprise beans are created in J2EE applications. Enterprise applications defined in the WebSphere Commerce 5.1 instance are mapped to J2EE applications with the same name and deployed under the default server; any other Web resources and beans that are mapped but not a part of an enterprise application are mapped into a default J2EE application. Web applications are mapped to WAR files; enterprise beans are deployed as EJB 1.1 beans in JAR files. These are combined into a J2EE EAR file and deployed on the new WebSphere Application Server platform.

Other infrastructure considerations

In addition to the upgrade to WebSphere Application Server Version 4 and, of course, WebSphere Commerce Business Edition Version 5.4, a site will need to upgrade the Web server running in the environment and, possibly, the database software being used. The software on which WebSphere Commerce Business Edition 5.4 is certified is updated regularly as new editions of the underlying software is released. In order to find the most recent information in this regard, please refer to:

http://www.ibm.com/software/webservers/commerce/wc_be/support.html

The above link to the IBM WebSphere Commerce Business Edition Version 5.4 support page also contains useful information about hardware requirements relating to processor and memory requirements.

14.2.2 Architecture changes

There have been changes made to the architecture of the product between WebSphere Commerce 5.1 and WebSphere Commerce Business Edition 5.4. Some of them require alterations in the way stores are assembled and handled.

WebSphere Commerce catalog

WebSphere Commerce 5.4 requires catalog data to meet a certain structure and, to that end, introduces the concept of a master catalog. The master catalog is a central location to manage a store's merchandise. It contains all of the products, items, relationships and pricing for everything in the store. Every store is required to have a master catalog.

WebSphere Commerce access control

Access control in WebSphere Commerce 5.1 was handled programmatically. In an effort to increase the customizability of access control in the commerce system, WebSphere Commerce Version 5.4 externalizes access control from the business logic and implements a policy-based model. Hierarchical access control is built into that methodology.

WebSphere Commerce member subsystem

In another departure from past release, every user and organizational entity in WebSphere Commerce Business Edition Version 5.4 must have a parent member which is another organizational entity; users and organizational entities can use this implementation to form a membership hierarchy. Member groups, notably, do not have parent members.

The default roles that are shipped in WebSphere Commerce 5.4 have changed a bit from earlier versions. Some new roles have been added and some removed as summarized in the following list:

- ▶ The Order Clerk role is not included as it is no longer required. The tasks performed by this role have either been automated or can be performed by other roles in WebSphere Commerce 5.4.
- ▶ The Customer role is not included in WebSphere Commerce Business Edition Version 5.4. It has been replaced by the AllUsers member group. Just as in WebSphere Commerce 5.1 the Customer access group is associated with a subset of commands that all users can execute, an access control policy has been set up to associate the subset of commands that can be executed by all users with the AllUsers member group in WebSphere Commerce 5.4. During data migration from WebSphere Commerce 5.1 to WebSphere Commerce Business Edition 5.4, any user explicitly assigned to the Customer access group in the former product will be explicitly assigned to the AllUsers member group in the latter.
- ▶ The Merchant role in WebSphere Commerce Version 5.1 has been renamed to Seller. The term Merchant is more closely associated with the business-to-consumer model while Seller is more in line with business-to-business.
- ▶ The Merchandising Manager role in WebSphere Commerce Version 5.1 has been renamed to Product Manager. The term Merchandising Manager is more closely associated with the business-to-consumer model while Product Manager is more in line with business-to-business.

14.2.3 Database changes

WebSphere Commerce Business Edition Version 5.4 contains many changes to the commerce database schema from earlier versions. In addition to alterations to the schema, there are other items to be cognizant of concerning the database during a migration from WebSphere Commerce Version 5.1 to WebSphere Commerce Business Edition Version 5.4. In this section, we review some of the more important changes in this area.

Commerce database schema

The commerce database schema has changed from WebSphere Commerce Version 5.1 to WebSphere Commerce Business Edition Version 5.4. For a complete list of the schema changes, please refer to the *What's New in IBM WebSphere Commerce V5.4* and *WebSphere Commerce Database Schema* documents available at:

http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

As an important example of the database schema changes from prior versions, the following tables are considered to be obsolete in WebSphere Commerce Business Edition 5.4:

- ▶ CMPGNINTV
- ▶ CMPGNRV
- ▶ INTVMPE
- ▶ INTVSGMT
- ▶ MAFAMILY
- ▶ MATYPE
- ▶ MPE
- ▶ MPETYPE
- ▶ ONQUEUE
- ▶ ONLOG
- ▶ ONSLOG
- ▶ ORDERMSG
- ▶ SEGMENT
- ▶ ZIPCODE
- ▶ ACCCMDGRP
- ▶ ACCMBRGRP
- ▶ ACCCUSTEXEC

National language support

If a national language version of WebSphere Commerce 5.1 was used, make sure that the LANGUAGE_ID for the default store is set correctly to the appropriate language in the STORE table and the populatedb script prior to the migration.

Database constraints

Often, stores customize tables which contain foreign keys to the standard WebSphere Commerce tables. During the data migration, the scripts may try to drop these referential integrity constraints. Foreign keys, primary keys, indexes and the like need to be dropped prior to migration in order to permit the script to complete successfully. Make note of the constraints that are dropped as they will need to be restored following the data migration.

Data premigration script

Before the commerce databases are migrated to the new WebSphere Commerce Business Edition Version 5.4 schema, the database preparation script must be run. This script will modify the access control tables so that they may be migrated correctly. It will also examine the existing commerce databases and generate a log about any item that does not have a parent product or any member that does not have a parent in the organizational entity. The log will reflect both mandatory and optional items that have been flagged by the script. These items should be fixed prior to data migration.

Examples of mandatory database fixes are outlined below.

- ▶ The default values for the members IDs in the WebSphere Commerce Version 5.1 database must be preserved for a successful database migration. These member IDs (0 to -8) must correspond to the correct member group (for instance, -7 must correspond to the Store Developer member group).
- ▶ In the AUCTION table, there is a REFCODE field on which WebSphere Commerce Version 5.1 did not enforce uniqueness. This field must be a unique index for WebSphere Commerce 5.4.
- ▶ There are fields whose maximum length has changed from WebSphere Commerce 5.1 to WebSphere Commerce 5.4. The lengths of these fields are checked.

Examples of optional database fixes are outlined below. While optional fixes need not be fixed immediately, it is recommended that these values be changed as time permits.

- ▶ WebSphere Commerce Business Edition Version 5.4 requires that each item have a parent product. If the store administrator does not create a parent product for an item which is flagged in the report generated by the premigration script, the database migration script will create one. Alternatively, if an item is found to have more than one parent product, then all parent products but one should be removed from the item.
- ▶ Any user assigned to the Customer access group (-2) in the WebSphere Commerce 5.1 database will automatically be assigned to the AllUsers

member group in WebSphere Commerce 5.4. Similarly, if any user is assigned to the default Order Clerk role (-5) in WebSphere Commerce 5.1, the role will be migrated as a user-defined role; the Order Clerk role does not exist as a default member group in WebSphere Commerce 5.4. If a store has no need for the Order Clerk role, remove any users from it prior to migration.

- ▶ If a store has created custom discount data manually (outside of the Merchandise menu in WebSphere Commerce Accelerator), the migration script will bring the data into the new database, but the WebSphere Commerce Business Edition 5.4 discount tool will be unable to see the discount.

14.2.4 Additional considerations

There are many other important things to consider when migrating to WebSphere Commerce Version 5.4. In this section, we review some of the more noteworthy among those considerations.

If any of the store's customized code used the `com.ibm.util` package in WebSphere Commerce 5.1, the code will need to be rewritten to use the equivalent class from the Java software developer's kit that is shipped with WebSphere Application Server Version 4 as the `com.ibm.util` package has been removed from current releases of WebSphere Application Server.

The lengths of the following columns have been changed from previous versions of WebSphere Commerce. Ensure that no data will be lost during migration by checking to see that no data contained therein is longer than the new length

- ▶ `MBRGRP.DESCRPTION` is now type `VARCHAR` and cannot exceed 512 characters.
- ▶ `ORGENCY.DESCRPTION` is also type `VARCHAR` with a limit of 512 characters,
- ▶ `CONTRACT.NAME` is now type `VARCHAR` and cannot exceed 200 characters.

It is recommended that all orders and order items have a status of C (indicating that payment has been authorized) that have truly been completed and shipped be changed to have a status of S (indicating that the order item has been shipped) in order to work with orders and order items using the WebSphere Commerce Accelerator tools.

The `EntityAdmin` and `HTTPCommandContext` commands from WebSphere Commerce Version 5.1 have been deprecated in WebSphere Commerce Version 5.4.

When migrating an instance, the WebSphere Commerce rules server is enabled by default regardless of its status in the WebSphere Commerce Suite 5.2 environment. In order to disable the rules server in WebSphere Commerce Business Edition Version 5.4 after the instance is migrated, change the enable directive from `true` to `false` in the `instance_name.xml` file.

While it is recommended that a store not alter bootstrap files, it may be necessary to do so at times. Any changes made to bootstrap values for the default store in WebSphere Commerce Version 5.1 should be reapplied after the migration to WebSphere Commerce Business Edition Version 5.4. During the migration, the default bootstrap data is loaded; this process overwrites any customizations a store may have made to the files. The bootstrap data for non-default stores is left untouched and should not need to be altered after migration.

The commerce system's database schema must be migrated prior to the migration of the WebSphere Commerce instance. Additionally, the script that performs the database migration should only be run once on a database; do not rerun the script on a database has already been migrated.

WebSphere Commerce Business Edition 5.4 requires that the distinguished name column be populated in the `ORGINITY` and `USERS` tables. Scripts are provided to fill these tables appropriately.

Any store migrated from a WebSphere Commerce 5.1 system will not have an `index.jsp` as it may have previously by default. In order to launch a store in this way, an administrator needs to create an `index.jsp` file. There are also several changes that need to be made to the store's JSPs in order to make them work in WebSphere Commerce Version 5.4. A tool, `migrateJSP`, is provided for this operation, but it should not be run against a file more than once as syntax errors may result.

SSL must be enabled on the Web server. Also, any customizations that were made to the Web server configuration in the WebSphere Commerce 5.1 environment must be reapplied if necessary.

If an administrator wants to use the staging server in WebSphere Commerce Business Edition 5.4, the Stage Copy utility needs to be run prior to the data migration in order to keep the production and staging databases consistent. Following the data migration, the staging server will need to be reconfigured.

14.3 Migration example(s?)

<Bill, if you are going to include an example or examples (please adjust the title of the section based on the number of examples), please insert them here; if there will not be any examples, please remove this section and edit the opening page text of this chapter to remove the reference to them. I would have written something to surround them, but I don't know what the topics are.>



A

Sample level 1 “a.” appendix heading (yHead0Appendix)

This appendix provides/describes/discusses/contains ...

In this appendix we provide/describe/discuss ...

In this appendix:

In this appendix, the following are described:

This appendix provides/describes/discusses/contains the following:

- ▶ ...
- ▶ ...
- ▶ ...
- ▶ Sample level 2 appendix heading
(created by **Special > Cross-Reference > Format: Head > Insert**)
- ▶ Sample next level 2 appendix heading

Sample level 2 heading (yHead1Appendix), new page

Add text here (Body0).

Sample level 2 heading (yHead2Appendix)

Add text here (Body0).

Sample level 3 heading (yHead3Appendix)

Add text here (Body0).

Sample level 4 heading (yHead4Appendix)

Add text here (Body0).

Sample level 5 heading (yHead5Appendix)

Add text here (Body0).



B

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/SG24????>

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, SG24????.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

<i>File name</i>	<i>Description</i>
?????????.zip	????Zipped Code Samples????

?????????.zip ???Zipped HTML Documents???

?????????.zip ???Zipped Presentations???

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: ???MB minimum???

Operating System: ???Windows/UNIX???

Processor: ??? or higher???

Memory: ???MB???

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.

Abbreviations and acronyms

NVP	Name value pair	JMS	Java Message Service
API	Application Programming interface	JNDI	Java Naming and Directory Interface
BMP	bean-managed-persistence	JNLP	Java Network Launching Protocol
CMP	container-managed-persistence	JSP	JavaServer Pages
CTS	Compatibility Test Suites	JTA	Java Transaction API
DoS	Denial of Service	JTS	Java Transaction Service
EAR	Enterprise Application Archive	LDAP	Lightweight Directory Access Protocol
EIS	Enterprise Information Systems	RMI	Remote method invocation
EJB	Enterprise JavaBean	RMIC	RMI Compiler
EJB-QL	EJB Query Language	WLQL	WebLogic Query Language
ENC	Environment naming context	XMI	XML Metadata Interchange
HTML	HyperText Markup Language	XML	eXtensible Markup Language
HTTP	HyperText Transfer Protocol	RFQ	Request for quote
IBM	International Business Machines Corporation	ERP	Enterprise resource planning
IIOP	Internet Inter-ORB Protocol	CRM	Customer relationship management
ITSO	International Technical Support Organization	B2C	business-to-consumer
J2EE	Java 2 Platform, Enterprise Edition	B2B	business-to-business
J2SE	Java 2 Platform, Standard Edition	CICS	Customer Information Control System
JAAS	Java Authentication and Authorization Service	ECI	External call interface
JAF	Java Activation Framework	SKU	Stock keeping unit
JAXM	Java API for XML Messaging	AAT	Application assembly tool
JAXP	Java API for XML Processing	SAR	Store archive file
JCA	J2EE Connector architecture	SCM	Source configuration management
JDBC	Java database connectivity	RAM	Random access memory
JDK	Java Development Kit	XPath	XML Path 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 417.

- ▶ *B2B e-commerce Using WebSphere Commerce Business Edition Patterns for e-business Series*,SG24-6180
- ▶ *e-commerce Patterns for Building B2C Web Sites Using WebSphere Commerce Suite V5.1*,SG24-6180
- ▶ *WebSphere Commerce Suite V5.1 Customization and Transition Guide*, SG24-6174
- ▶ *WebSphere V3.5 Handbook*, SG24-6161
- ▶ *WebSphere Version 4 Application Development Handbook*, SG24-6134
- ▶ *IBM WebSphere V4.0 Advanced Edition Handbook*, SG24-6176
- ▶ *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167
- ▶ *WebSphere Commerce V5.4 Handbook Architecture and Integration Guide*, SG24-6567
- ▶ *DB2 UDB V7.1 Performance Tuning Guide*, SG24-6012
- ▶ *Java Connectors for CICS: Featuring the J2EE Connector Architecture*, SG24-6401
- ▶ *VisualAge for Java Enterprise Version 2 Team Support*,SG24-5245

Other resources

These publications are also relevant as further information sources:

- ▶ *The Elements of Java Style*, Vermeulen, Ambler et al. Cambridge University Press, ISBN 0-521-77768-2
- ▶ *E-Commerce User Experience*, Jakob Nielsen, Rolf Molich et al. Nielsen Norman Group, ISBN: 0-9706072-0-2
- ▶ *????full title????????, xxxx-xxxx*

Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ Concurrent Versions System
<http://www.cvshome.com>
- ▶ Writing Robust Java Code
<http://www.ambysoft.com/javaCodingStandards.pdf>
- ▶ Jtest home page
<http://www.parasoft.com/jsp/products/home.jsp?product=Jtest>
- ▶ Junit home page
<http://www.junit.org>
- ▶ Segue software
<http://www.segue.com>
- ▶ WinRunner test tool
<http://www.mercuryinteractive.com/products/winrunner>
- ▶ Jinsight page
<http://www.alphaworks.ibm.com/tech/jinsight>
- ▶ Page Detailer documentation
<http://ibm.com/software/webervers/studio/doc/v35/pagedetailer/EN/HTML>
- ▶ Design for performance
<http://www7b.software.ibm.com/wsdd/library/techarticles/hvws/perform.html>
- ▶ NetIQ's Security Analyser
<http://www.Webtrends.com/products/wsa>
- ▶ Tivoli Intrusion Manager
http://www.tivoli.com/products/index/intrusion_mgr/
- ▶ Computer Associates eTrust Intrusion Detection
<http://www3.ca.com/Solutions/Solution.asp?ID=271>
- ▶ Mercury Interactive ActiveTest SecureCheck
<http://www-svca.mercuryinteractive.com/products/securecheck/>
- ▶ Apache Ant
<http://jakarta.apache.org/ant/index.html>
- ▶ Apache Ant manuals
<http://jakarta.apache.org/ant/manual/index.html>

- ▶ WebSphere Commerce Business Edition library
http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html
- ▶ CICS home page
<http://www.ibm.com/cics>
- ▶ CICS Transaction Gateway documentation
<http://www-3.ibm.com/software/ts/cics/library/manuals/eindex40.htm>
- ▶ WebSphere Commerce family page
<http://www.ibm.com/software/webservers/commerce/>
- ▶ Improve Web site performance with Page Detailer
<http://www7.software.ibm.com/vad.nsf/data/document4361>
- ▶ WebSphere Developer Domain search
<http://www7b.software.ibm.com/wsdd/library/>
- ▶ WebSphere Developer Domain library for VisualAge for Java
<http://www7.software.ibm.com/vad.nsf/Data/Document2001>
- ▶ Writing JMS programs using MQSeries and VisualAge for Java
<http://www-106.ibm.com/developerworks/ibm/library/it-farrell11/index.html>
- ▶ Xmlspy
<http://www.xmlspy.com/>
- ▶ XML Path Language
<http://www.w3.org/TR/xpath>

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

R

Redbooks Web site 417
Contact us xiii



WebSphere Commerce V5.4 Developer's Handbook

(0.5" spine)
0.475" x 0.875"
250 <-> 459 pages



Redbooks

WebSphere Commerce V5.4 Developer's Handbook

Understanding the development process

This redbook details the new tools and techniques available to J2EE developers using WebSphere Commerce V5.4 to customize e-commerce shopping sites.

Planning and using a development environment

It provides worked examples of customizations proceeding from realistic requirements through to modification of store assets, including front end, data and back end assets.

Customization examples

Customization examples include both simple customization and complex programming customizations

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks